1992年8月1日発行（毎月1回1日発行）第11巻8号通巻124号　昭和58年11月2日第三種郵便物認可

PERSONAL COMPUTER MAGAZINE for MZ, X1, and X68000

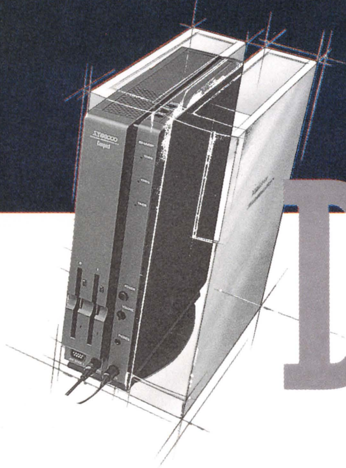
Oh!X

SOFT BANK オー！エックス
定価600円

特集 プログラミング再入門

全機種共通システム SLANG用MAGIC対応グラフィックライブラリ
新製品紹介 グラフィックツールMATIER/SOUND SX-68K/MIDI音源TG100
続々・創刊10周年PRO-68K/続々・創刊10周年特別企画 連載のすべて（後編）

8
1992



DownSizing

X68000 見・体・験フェア inサマー

あのX68000アイドル山下章氏をむかえておくる、アツイアツイ夏祭り。X68000ユーザーはもちろん、パソコンならおまかせのキミ、そしてパソコン未体験の人も、JOIN WITH US!

- ●「第1回全日本X68000芸術祭・全国大会」ビデオ上映
- ●新作ソフト紹介……などなど見逃せない内容がイッパイ。

資料請求券
X68000
Oh!X
8月

体積比44%(当社従来比)、このサイズが象徴するのはまさに創造力とテクノロジーの無限大の可能性です。この先、X68000がどう発展していくのか、その夢の一端が、コンパクトなボディに託されています。ベーシックにはX68000そのもの、しかし未来に夢を結ぶユーザーインターフェイスやデバイスを新たに搭載。はじめて触れる人には、優しさで迎えます。もっと追求したい人には、賢さで応えます。何かを生み出したい、自分を表現したい、誰もが抱く「創造力の芽」をひとりひとりの個性に合わせて大きく育む。そんな夢工房がここにあります。

無限大の可能性は そのままに、 そのサイズだけを 凝縮しました。

この事実はX68000の未来に、さらなる可能性をひらくことになるだろう。

●X68000のさらなる夢を象徴する体積比44%(当社従来比)のコンパクトサイズ●成熟するウィンドウ環境、SX-WINDOW ver.2.0搭載:フォントマネージャーを装備してアウトラインフォントに対応/1024×1024ドットのワイドデスクトップ、画面スクロールによる軽快なハンドリングをサポート/アイコンの作成・編集を可能にするパターンエディタ&アイコンメンテ/ポップアップメニューを自在に作成できるメニューメンテ/ディレクトリ構造やファイル情報を一覧表示できるツリービューア/その他クリップボード、シンボルトレイなどユーザーインターフェイスを高める新機能を装備●2HD3.5インチFDD2基搭載●カラー液晶ディスプレイとも接続可能※●マウス、コンパクトキーボード標準装備●16MHzクロックをはじめ、X68000XVIの機能を継承。 ※カラー液晶ディスプレイを接続してご使用の場合、SX-WINDOW上のアプリケーション利用に限定されます。

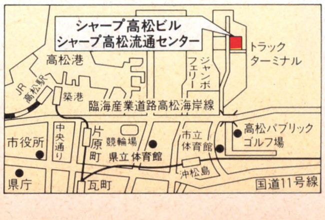
●10.4型TFTカラー液晶ディスプレイ
LC-10C1-H(グレー) 標準価格598,000円(税別)
●接続ケーブル
AN-1515X 標準価格4,200円(税別)

New
X68000
PERSONAL WORKSTATION・XVI
Compact

本体+キーボード+マウス
2HD3.5インチFDDタイプ CZ-674C-H(グレー) 標準価格298,000円(税別)
14型カラーディスプレイ(ドットピッチ0.28mm)
CZ-608D-H(グレー) 標準価格94,800円(税別)

●5.25インチ増設用フロッピーディスクドライブ CZ-6FD5 標準価格99,800円・税別〔接続ケーブル同梱〕
●ディスプレイテレビ/CZ-6TU用RGBケーブル CZ-6CR1 標準価格4,500円・税別
●ディスプレイテレビ/CZ-6TU用テレビコントロールケーブル CZ-6CT1 標準価格5,500円・税別
●SCSI変換ケーブル CZ-6CS1 標準価格12,000円・税別

開催日時:7月26日(日)13:00〜17:00
会　場:シャープ高松ビル5Fホール(ジャンボフェリーのりば北300m)
高松市朝日町6-2-8 ☎0878-23-4860(代)

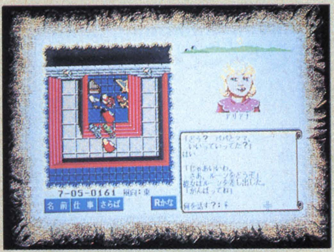


■主催・お問い合わせ/シャープエレクトロニクス販売(株) 四国統轄(営) ☎0878-23-4860(代) 担当・細川

●お問い合わせは…
シャープ株式会社
電子機器事業本部システム機器営業部
〒545 大阪市阿倍野区長池町22番22号
☎(06)621-1221(大代表)
電子機器事業本部AVCシステム事業推進室
〒162 東京都新宿区市谷八幡町8番地
☎(03)3260-1161(大代表)

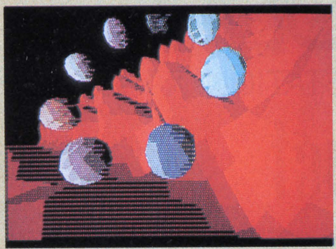
MATIER

GM対応音源 TG-100

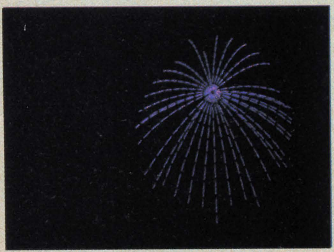


ウルティマVI

三國志III



グラフィックライブラリ



(影)のショートプロぱーてぃ

Oh!X

CONTENTS

●特集

81 プログラミング再入門

82	まず本質を知る プログラミング言語の前に	中森 章
84	フローチャートによるアイデアのまとめ プログラムの流れをつかもう	文月 涼
90	BASICで作るDōGAフレームファイル 正しい花瓶の落とし方	柴田 淳
98	オブジェクト指向に学ぶ 作り散らかせます	丹 明彦
105	比較的大きなプログラムの独断的制作法 ちょっと大きいモノを書こう	横内威至

●続々・創刊10周年記念PRO-68K

49	Z-MUSIC ver.1.10	西川善司
56	ゲーム内部のイロハ	浜崎正哉
60	LIFE110.X	石川淳二

●続々・創刊10周年特別企画「Oh!MZ.Oh!X 10年間の歩み」

63 連載のすべて（後編）

●カラー紹介

20	新製品紹介 MATIERを使う(前編)	中野修一

●THE SOFTOUCH

22	SOFTWARE INFORMATION 新作ソフトウェア/TOP10	
24	TREND ANALYSIS	
	GAME REVIEW	
26	ウルティマVI	瀧 康史
28	三國志III	金子俊一
30	バトルテック	影山裕昭
32	シムアース	荻窪 圭
34	AFTER REVIEW グラディウスII	

〈スタッフ〉
●編集長／前田 徹 ●副編集長／植木章夫 ●編集／岡崎栄子 浅井研二 山田純二 ●協力／有田隆也 中森 章 林 一樹 吉田幸一 華門真人 吉田賢司 影山裕昭 大和 哲 村田敏幸 丹 明彦 三沢和彦 長沢淳博 宮島 靖 金子俊一 浦川博之 石上達也 柴田 淳 御木徳高 ●カメラ／杉山和美 ●イラスト／永沢しげる 山田晴久 寺尾響子 ●アートディレクター／島村勝頼 ●レイアウト／元木昌子 ADGREEN ●校正／グループごじら

表紙絵：須藤　牧人

E　N　T　S

●シリーズ全機種共通システム

111 THE SENTINEL

112 実践Small-C講座(5)ワイルドカード　石上達也

116 グラフィックライブラリ GRAPH.LIB　黒木淳一

●読みもの

158 猫とコンピュータ　第72回
新説・猫×7×0.7　高沢恭子

160 X-OVER NIGHT　第25話
IKEBUKURO　高原秀己

162 第62回　知能機械概論ーお茶目な計算機たちー
なぜ13分で料理が消えたのか　有田隆也

●連載/紹介/講座/プログラム

18 響子 in CG わ〜るど［第15回］
羽　寺尾響子

36 新製品紹介
SX-WINDOW対応音色エディタ SOUND SX-68K　紀尾井誠

38 吾輩はX68000である［第14回］
飛び出せ！　ディスク　泉　大介

42 新製品紹介
GM対応音源モジュール TG100　高橋哲史

45 ハードウェア工作入門〈26〉コンピュータアーキテクチャ編
論理演算で加算器を作る　三沢和彦

66 DōGA・CGアニメーション講座
みんな準備はいいか？　かまたゆたか＆MAX田口

68 よいこのSX-WINDOW講座（第9回）
イメージを極める　中森　章

125 Creative Computer Music入門（11）
効率的な採譜のやり方　瀧　康史

133 X68000マシン語プログラミング Chapter_22H
スプライトを使いこなす　村田敏幸

140 Oh!X LIVE in '92
氷穴（X68000・Z-MUSIC/PCM8.X用）　上田浩司
ガラガラヘビがやってくる（X68000用）　祢津伸也
風の贈り物（X1/turbo用）　長坂和彦

148 大人のためのX68000［第22回］
パソコン通信に未来はあるか　荻窪　圭

152 (影)のショートプロぱーてぃ　その35
夏です，金鳥です，花火です　影山裕昭

156 ANOTHER CG WORLD　寺尾響子

ペンギン情報コーナー……164
FILES Oh!X……166
Oh!X質問箱……168
STUDIO X……170
編集室から/DRIVE ON/ごめんなさいのコーナー/SHIFT BREAK/microOdyssey……174

1992 AUG.
8

UNIXはAT & T BELL LABORATORIESのOS名です。
Machはカーネギーメロン大学のOS名です。
CP/M, P-CPM, CP/Mplus, CP/M-86 CP/M-68K, CP/M-8000, DR-DOSはデジタルリサーチ
OS/2はIBM
MS-DOS, MS-OS/2, XENIX, MACRO80, MS C, MS-WindowsはMICROSOFT
MSX-DOSはアスキー
OS-9, OS-9/68000, OS-9000, MW CはMICROWARE
UCSD p-systemはカリフォルニア大学理事会
TURBO PASCAL, TURBO C, SIDEKICKはBOLAND INTERNATIONAL
LSI CはLSI JAPAN
HuBASICはハドソンソフト
の商標です。その他，プログラム名,CPUは一般に各メーカーの登録商標です。本文中では“TM”,“R”マークは明記していません。
本誌に掲載されたプログラムの著作権はプログラム作成者に保留されています。著作権上，PDSと明記されたもの以外，個人で使用するほかの無断複製は禁じられています。

■広告目次

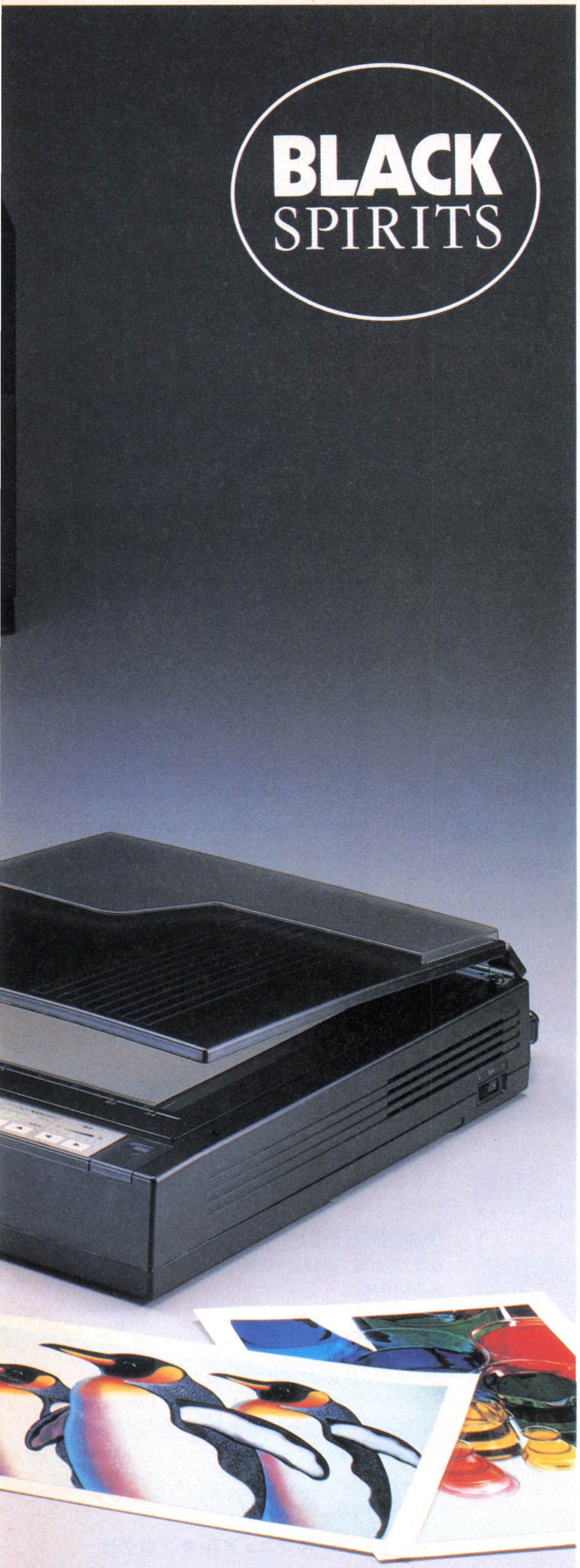
アイビット電子……181
アクセス……184
エニックス……12
カプコン……10
計測技研……182
サン・ミュージカル・サービス……11
Ｊ＆Ｐ……表3
シャープ……表2・表4・1・4-7
九十九電機……13
パソコンプラザオクト……178・179
ビクター音楽産業……9
Ｐ＆Ａ……14・15
ブラザー工業……8
マイクロウェアシステムズ……183（上）
マイコンショップ川口……180
満開製作所……177
ラインシステム……183（下）

SHARP

シャープ株式会社 〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)

入力から、編集、出力、複写まで、カラーの世界は、シャープです。

SHARP IS COLOR



▶INPUT

X68000用パラレルインタフェースを標準装備した高速コンパクト型イメージスキャナ。

カラーイメージスキャナ JX-220X……標準価格168,000円(税別)

●A4サイズの原稿を約50秒※1で高速読み取り●CCDセンサー採用。さらに中間調処理でシャープでリアルな画像を再現●ディザパターン指定機能※2や濃度補正機能※2など高度な画像処理機能で緻密な読み取りが可能●解像度200ドット/インチ(約7.9ドット/mm)。ズーム機能で1%きざみの拡大、縮小も可能●色ずれの少ない線順次(1走査)読み取り●X68000シリーズ用「スキャナツール」ソフトを標準装備●プリンタと直接接続することによりダイレクトプリント※3が可能●RS-232Cインタフェース/X68000シリーズ用専用パラレルインタフェースを標準装備。

※1：A4、2値出力、コンピュータへの実転送時間。
※2：表記機能はJX-220X本体使用であり、付属ユーティリティ使用時は異なります。
※3：別売のパラレルインタフェースケーブル(JX-22PC標準価格12,000円(税別)が必要です。

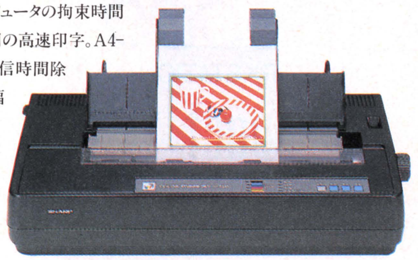
▶OUTPUT

3種類の制御コマンドモードを搭載。質感も鮮やかに再現する高品位カラーイメージジェット。

カラーイメージジェット IO-735X-B……標準価格248,000円(税別)

●シャープ独自のIOシリーズコマンド(Gモード)に加え、NM-9900モード(Nモード)、ESC/P24-84C準拠モード(Pモード)をサポート。一般文書の作成から、各種デザイン、建築用パースなどのCAD分野に対応●発色性に優れた普通紙対応の新黒インキ採用。専用紙はもちろんオフィスでよく使われる普通紙にも鮮明カラー印字●プリントバッファメモリ(128KB)の内蔵で、ホストコンピュータの拘束時間を軽減●48ノズル(各色12ノズル)採用の高速印字。A4-1ページを※約90秒でプリント(データ受信時間除く)●ビジネス用途に適したB4横用紙幅対応●OHPフィルム(専用)にも鮮明プリント●ノンインパクト方式ならではの静粛印字●インキ補充は簡単、経済的なカートリッジ方式

※261×174mm領域



IO-735X-B 対応アプリケーション

●SX-WINDOW対応ペイントツール
Easypaint SX-68K
CZ-263GW 標準価格12,800円(税別)

●WYSIWYGを実現、ドローグラフィックソフト
CANVAS PRO-68K
CZ-249GS 標準価格29,800円(税別)

●オリジナリティを活かせるポップアップツール
NEW Printshop PRO-68K ver2.0
CZ-221HS 標準価格20,000円(税別)

●マルチワープロ PRO-68K
Multiword
CZ-225BS 標準価格32,000円(税別)

●高速カード型リレーショナルデータベース
CARD PRO-68K ver2.0
CZ-253BS 標準価格29,800円(税別)

●パソコン通信もできるメモリ常駐型ソフト
Teleportion PRO-68K
CZ-258BS 標準価格22,800円(税別)

●これからの高速通信をサポート
Communication PRO-68K ver2.0
CZ-257CS 標準価格19,800円(税別)

資料のご請求・お問い合わせはコンシューマーセンター／東日本相談室 〒162 東京都新宿区市谷八幡町8番地 ☎(03)3260-1161(大代表)・西日本相談室 〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)・名古屋相談室 〒454 名古屋市中川区山王3丁目5番5号 ☎(052)332-2611(大代表)・福岡相談室 〒816 福岡市博多区井相田2丁目12番1号 ☎(092)575-2381

SHARP

目の付けどころが、
シャープでしょ。

待望のSX-WINDOW
開発支援ツール、登場。

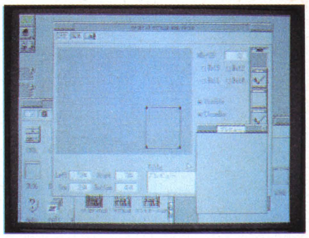
NEW

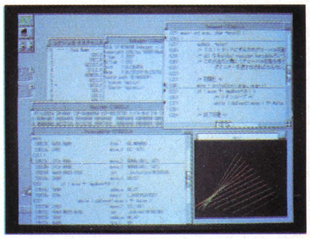
SX-68K

SX-WINDOW 開発キット

CZ-288LWD 9月発売予定

SX-WINDOW用のソフト開発に必要な開発ツールやサンプルプログラムを装備。プログラムの編集、リソースの作成、コンパイル、デバッグといった一連の作業をSX-WINDOW上で効率よく実行できます。初めてSX-WINDOW用のプログラムに挑戦する人にも、簡単に基本機能の理解ができる33種のサンプルプログラム付き。また各マネージャ解説と関数リファレンスの詳細なマニュアルも装備しています。





※本ソフトのご使用に際しては、メインメモリ4MB以上、SX-WINDOW ver2.0以上、C compiler PRO-68K ver2.0以上が必要です。

キット構成

■開発ツール

●SXデバッガ
SX-WINDOW上で複数のプログラムを同時にデバッグすることができるソースコードデバッガ。

●リソースエディタ
SX-WINDOW上のリソースをリソースタイプごとの編集ウィンドウでビジュアルに作成・編集が可能。

●リソースリンカ
Cコンパイラやアセンブラで作成したリソースデータファイル(オブジェクトファイル)をリンクしてリソースファイルを作成。

●サンプルメイク
サンプルプログラムのコンパイル作業をSX-WINDOW上から、XC ver2のMAKE. Xを呼び出して、自動実行する簡易メイクユーティリティ。

■サンプルプログラム

●基礎編(23種)
各マネージャの基本的な機能のみを用いた基本動作の理解。

●応用編(4種)
基礎編での基本機能を応用した簡単なアプリケーションの作成。

●実用編(6種)
基礎/応用編での機能を駆使した、実用的なアプリケーションの作成。

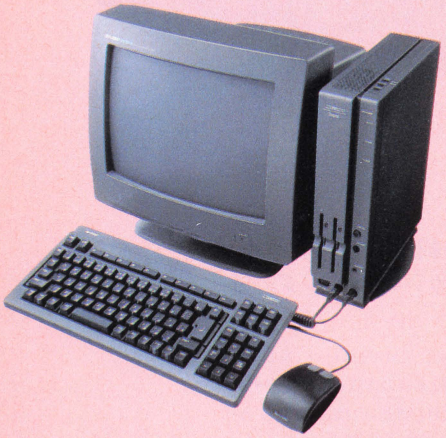
■その他のファイル

●インクルードファイル
Cコンパイラとアセンブラ用の関数定義、データ定義ファイル。

●ライブラリファイル
Cコンパイラ用の関数ライブラリ。

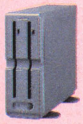
マニュアル

●ユーザーズマニュアル ●プログラマーズマニュアル ●ファンクションリファレンス ●ライブラリリファレンス



X68000
PERSONAL WORKSTATION・XVI
Compact

本体＋キーボード＋マウス
2HD3.5インチFDDタイプ
CZ-674C-H(グレー) 標準価格298,000円(税別)
14型カラーディスプレイ(ドットピッチ0.28mm)
CZ-608D-H(グレー) 標準価格94,800円(税別)



●5.25インチ増設用フロッピーディスクドライブ
CZ-6FD5
標準価格99,800円・税別
〔接続ケーブル同梱〕

●ディスプレイテレビ/CZ-6TU用RGBケーブル
CZ-6CR1 標準価格4,500円・税別
●ディスプレイテレビ/CZ-6TU用テレビコントロールケーブル
CZ-6CT1 標準価格5,500円・税別
●SCSI変換ケーブル CZ-6CS1 標準価格12,000円・税別

資料請求券
CZソフト
Oh!X
8係

開いてくださいウィンドウ、触れてくださいインテリジェンス。
さらに拡がる、SXワールド。

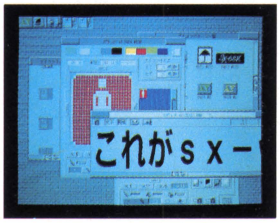
●アウトラインフォント対応、ひらかれたウィンドウ環境。

SX-WINDOW ver2.0

CZ-287SS 標準価格12,800円(税別)

フォントマネージャを装備して待望のアウトラインフォントに対応。画面スクロール機能により、表示画面よりワイドなデスクトップ空間を駆使。アプリケーションのハンドリングに便利なシンボルトレイやアイコンメンテ、パターンエディタなど便利機能満載。

※SX-WINDOW ver1.0(CZ-259SS)およびSX-WINDOW ver1.1(CZ-278SS)をお持ちの方には有償バージョンアップを行います。



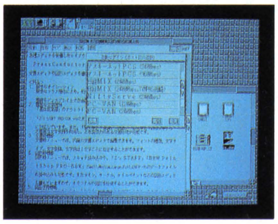
●マルチタスク機能をはじめ、通信環境がさらに充実。

Communication SX-68K

NEW

CZ-272CWD 7月発売予定

通信環境をさらに高めたウィンドウ対応の通信ソフトです。マルチタスク機能により他のアプリケーションソフトを実行中でも簡単に通信が可能。また、ホスト局をクリックするだけの自動ログイン機能、初心者にも簡単なプログラム機能、最新モデム(20種類)もフルサポートしています。



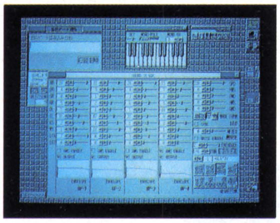
●多彩なサウンドクリエイトを実現するFM音源サウンドエディタ。

SOUND SX-68K

NEW

CZ-275MWD 8月発売予定

他のミュージックソフトで演奏中の音色を、簡単に作成・変更ができるマルチタスク機能、またエディット、イメージ、ウェーブの3つの編集/確認モードを装備。作成中の音色も50曲の自動演奏でリアルタイムに確認、編集できます。まさにミキサー感覚で音創りが楽しめるツールです。



●ウィンドウ対応グラフィックツール。

Easypaint SX-68K

CZ-263GWD 標準価格12,800円(税別)

マウスによる簡単操作、65,536色中16色の多彩な表現、クリエイティブマインドに応えるウィンドウ対応ペイントツールです。同時に複数のウィンドウを開いて編集でき、各ウィンドウ間でのデータ交換もできます。

※SX-WINDOW対応ソフトの動作には、メインメモリ2MBおよびSX-WINDOW ver1.1以上が必要です。

充実のPROシリーズ

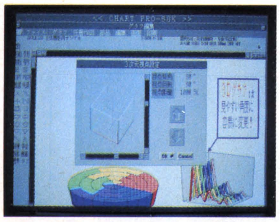
●ビジネスグラフチャート

CHART PRO-68K

NEW

CZ-267BSD 標準価格38,000円(税別)

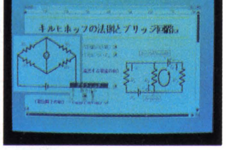
各種データベースで作成したデータをもとに、多彩なグラフが作成できます。3次元表示やグラフの複合機能も装備。データはMultiword,Press conductor PRO-68Kに取り込むこともできます。



●グラフィック機能搭載の本格派ワープロ

Multiword ver1.1

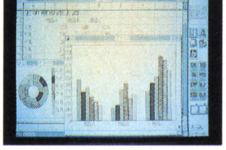
CZ-225BSD 標準価格32,000円(税別)



●簡単操作の統合型表計算ソフト

BUSINESS PRO-68K Popular

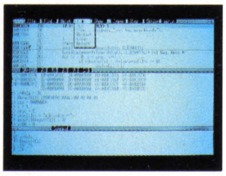
CZ-286BSD 標準価格28,000円(税別)



●各種ドライバ、ライブラリを追加

COMPILER PRO-68K ver2.1

CZ-285LSD 標準価格44,800円(税別)

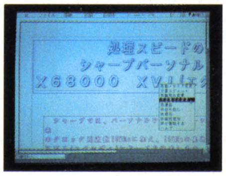


※有償バージョンアップ対応中。

●各種エディタ装備のレイアウトソフト

PressConductor PRO-68K

CZ-266BSD 標準価格28,000円(税別)



※以上のPROシリーズのソフトの動作にはメインメモリ2MB必要です。

※発売予定のソフトの画面写真は実物とは異なる場合があります。

●お問い合わせは…シャープ(株)電子機器事業本部AVCシステム事業推進室 〒162 東京都新宿区市谷八幡町8番地 ☎(03)3260-1161(大代表)へ。 シャープ株式会社

パソコンソフト自販機 TAKERU

brother

LIFE & DEATH ™

ライフ　デス

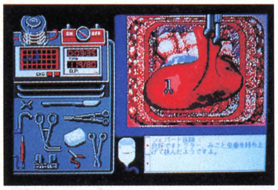
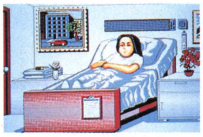
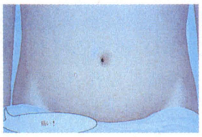
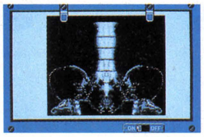
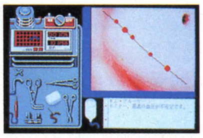
THE SOFTWARE TOOLWORKS

私達は、生命の神秘に出会った。

日本初登場!! 欧米で大ヒットの外科手術シミュレーションゲーム。

外科医だけに与えられた"手術"という領域を、アカデミックな表現と映像でシミュレートする
究極のメディカルゲーム「Life & Death」。このゲームであなたは、
人体の精緻と生命の神秘、そして生への真摯な眼差しに出会うことだろう。

Copyright ©1990 The Software Toolworks,Inc. All right reserved. The Software Toolworks and Life & Death are registered trademark of The Software Toolworks,inc.
©1992 Japanese version by VING CO., LTD.

好評発売中!!

価格(税込)￥7,000

■対応機種：X68000　■企画/開発：アローマイクロテックス　VING

超人 — Cho Jin —

●パワーアップアイテムをいかにつかむかが攻略の鍵。

●次々に現れる敵キャラを倒せばステージクリア。ボスキャラは当然、手強いぜ！

●大型銃とバリアだけのシンプルなルールが新しい。

●たった一度の被弾でもこうなってしまう。断末魔の叫び声と悲惨な死が待っている。

●X68000ならではの、迫力と鮮烈な画面。新しい興奮が、今経験できる。

斬新な上方見おろしの
ニュータイプ・フィールドバトルアクションゲーム
超人(Cho-Jin)。

ステージは全50面。フィールド上のモンスターや謎の殺人マシンを倒し、10面毎に現れるボスキャラに挑め！

走る！叫ぶ！うねる！
ADPCM同期100曲以上のビートの効いたBGMにのり、今超人は熱く燃える!!

好評発売中

価格(税込) ￥4,800

■対応機種：X68000　■企画/制作：fix

ニュース!! TAKERUでは、X68000コンパクトXVに対応した3.5"2HD版ソフトを発売！ただ今、TAKERUにのってるX68000タイトルは全て、3.5"2HD版があります!!

ブラザー工業株式会社　〒467 名古屋市瑞穂区苗代町2番1号

TAKERU事務局 (052)824-2493

東京営業所(03)3274-6916
大阪営業所(06)252-4234

通信販売　通信販売をご希望の方は、ソフト名・機種名・住所・氏名・電話番号を明記の上TAKERU事務局まで現金書留でお申し込み下さい。
代金引換は一度、現金書留で申し込んで頂いた方に案内させて頂きます。

Victor
BATTLETECH®
バトルテック
奪われた聖杯
奪われた聖杯を取り戻せ！広大な宇宙空間に展開されるロボット・ウォーズ！
3024年、惑星アンダースムーンのデューク、キャメロンが、何者かの策略により命を落とし、皇位継承のシンボル「ハーンの聖杯」も奪われてしまった　父の復讐と奪われた聖杯を取り戻すために、キャメロンの息子ギデオン・ブレーバーは広大な宇宙へと旅立つ。
好評発売中！
迫力の3Dポリゴン戦闘。前進、後退、ジャンプなどの動きと武器選択、照準移動、敵ダメージ表示などの多角的機能を駆使。
用意されたメック（ロボット）は機能色々の変化に富んだ全8体。
傭兵は最大3名。プレイヤー自身をふくめてプレイヤー以外に戦略をたてて敵のロボットを粉砕するのだ。
■アメリカで爆発的なロボットブームを巻き起こした話題作いよいよ登場！
■3Dポリゴンの採用による迫真のバトル・アクション
■プレイヤーが自らコックピットに乗り込みロボットを操縦、リアルなロボット・シミュレーションを体験
■共に戦うクルーとして41人の傭兵から最大3人までの傭兵の採用により、戦略性もゲームの重要な要素
■情報収集と賞金稼ぎによってグレイドの高いメックを手にいれて、150の惑星を舞台に任務を遂行
〈バトルテック〉シリーズは、実際に体験したり読んだりと様々なメディアで楽しめます
アミューズメントセンター「バトルテックセンター」
92年夏横浜にオープン（提供、株式会社シグマ）
小説、設定集などの関連書籍続々発売（発売、富士見書房）
© 1992 Activision. All rights reserved.
Battletech is registered trademarks of FASA Corporation.
© 1992 Victor Musical Industries, Inc.
発売元 ビクター音楽産業株式会社
通信販売 当社の商品をお近くのパソコンショップでお買い求めになれない場合、商品名、機種名、住所、氏名、電話番号を明記のうえ、下記住所まで定価プラス3％消費税分を現金書留にてお申し込み下さい（送料無料）〒151 東京都渋谷区千駄ヶ谷2-8-16 ビクター音楽産業㈱〈通信販売係〉

CAPCOM

受け継がれた
ファイティング・スピリット。

CAPCOM
X68000
ソフト
第1弾

Final Fight®

プラス！ スペシャルCD付

あの激闘のすべてが、
X68000版ファイナル ファイトで実現。

★二人同時プレイ可能!!

7月17日発売予定 定価9,800円
この価格には消費税は含まれておりません。

MIDI
MIDI音源対応

No Copy
このマークは
不法コピー
禁止マークです
ソフトウエア法的保護監視機構

MIDI音源対応(ローランド社MT-32、CM-64、CM-32L)MIDIを使用するには別売のMIDIボード(SHARP)CZ-6BM1、または(SYSTEM SACOM)SX-68Mが必要です。

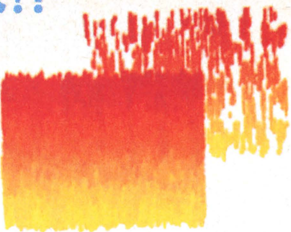
ジョイスティック対応

©CAPCOM 1990.1992 ALL RIGHTS RESERVED.

株式会社カプコン®
大阪本社営業／〒540 大阪市中央区釣鐘町2−2−8　東京支店／〒163-02 東京都新宿区西新宿2−6−1 新宿住友ビル43F
※カプコンソフト情報 大阪(06)946-6659 東京(03)3340-0718 札幌(011)281-8834 仙台(022)214-6040 名古屋(052)571-0493 広島(082)243-6264 松山(0899)34-8786 福岡(092)441-1991
◎電話番号は、おかけ間違いのないようにお願いします。

何がすごいか！

2次元なのに3次元!?



1 リンゴの模様を自動ペインティング機能で一気に作成します。

2 光源やハイライトを設定して球状に３Ｄ変形します。

3 最後に、メッシュコントロールによる自由変形機能で変形すれば、簡単にリアルなリンゴの完成です。

X68000がグラフィックワークステーションに変身!

マチエールは、プロのデザイナー集団がつくったプロのためのペイントソフトです。高い機能と使いやすいマウスオペレーションを実現しました。

大画面編集も思いのままに

512×512ドット標準画面の解像度では、フィルム出力をして印刷物にするには不足です。マチエールではメモリー増設により最大2048×2048ドットの画面をリアルタイムに編集できます。

複数の画面で快適編集

512×512ドットを同時に最大4画面までもつことができます。(メモリ2Mバイト時は2画面まで)絵のパーツを作っておいたり、2つの画面を合成したり、クリエイティブワークの能率が大幅アップします。画面間の便利な合成機能もいろいろ用意してあります。

立体文字の作成も簡単

どんな図形も簡単に立体表現することができます。「書体倶楽部」(Zeit社)のアウトラインフォントや、スキャナでとりこんだロゴマークなども、マチエールで立体文字にすれば、ビジュアル効果も抜群です。

ディザでフルカラーを実現

マッハバンドのない美しいグラデーションは、角度・増減率とも自由に設定可能、4隅の色指定もできます。
ぼかし・3次元表示など高度な画像処理も1670万色フルカラーで実現しました。

ジャギーのない高品質

拡大・縮少・変形・パース変形・メッシュ変形など、すべてオーサンプリングによるジャギーのない高品質を実現しました。

多彩な編集機能

コピー・クリップコピー・シェードコピー・タイルコピー・拡大・縮小・変形・回転・パース変形・メッシュ自由変形・円筒マッピング・球面マッピング・領域交換・矩形スクロール・ミラー反転・各種マスク機能

専用ソフトなみの画像処理機能

ネガ反転・ディフューズ・ぼかし・モノクロ化・二値化・ランダムノイズ・平滑化・鮮鋭化・輪郭抽出・レリーフ・モザイク・フレア・コントラスト補正・色変換など

使いやすいスキャナー入力

スキャナ原稿台のプレビュー表示をマウスで範囲指定する簡単操作。

高機能なプリンタ出力

画面の任意の範囲を、最大A３までの自由なサイズでプリントアウトできます。

手軽にCDアニメーションも

アニメーションツール「うごくZO」を標準でバンドルしました。手軽にCGアニメーションが楽しめます。

■ 対応画像フォーマット・入出力機器

画像ファイル形式	PIC・GL3・GLX・IMG・RGB・TIFF(Mac・TOWNS互換)
カラープリンター	SHARP IO-735X・SHARP CZ-8PC3・SHARP CZ-8PC5・NEC PC-PR406
ビデオプリンタ	SHARP CZ-6PVI・NEC PC-VC101
ビデオ取り込み	SHARP CZ-6VT1
カラースキャナー	SHARP CZ-8NS1・SHARP JX-220X(以上純正パラレルボード対応)・EPSON GT-4000・EPSON GT-6000

■ 監修　CGデザイナー　長谷川 一光

新登場

9月上旬発売予定

プロ仕様ペイントツール

Hyper Image Processor

Matier

商標登録申請済

マチエール

対応機種 X68000(要2M)　価格39,800円(税別)

株式会社 サンワード

SUNWARD

〒154 東京都世田谷区池尻3-21-28 池尻成和ビル TEL(03)3419-8839

ENIX
PROGRESSIVE
MOVEMENT
NOW ON SALE!
ENIX
EFX-140
X68000シリーズ対応
ハードディスクユニット
外付SCSIタイプ（ターミネータ付）
◆EFX-100B　標準価格（税別）118,000円
●容量　約100MB
●平均アクセスタイム　19㎜/sec
◆EFX-140B　標準価格（税別）138,000円
●容量　約140MB
●平均アクセスタイム　16㎜/sec
発売・販売元
ENIX 株式会社エニックス
〒160 東京都新宿区西新宿7-5-25
サポートセンター　TEL03-3367-1908
※デザイン・仕様等は改良を目的として予告なく変更する場合がございます。また表示価格には消費税は含まれておりません。ご了承下さい。
※お求めは全国のパソコンショップ等で。

親切と安さで信頼のあるTSUKUMO

掲載商品2万円以上送料無料（一部地域を除く）

期間内のみの限定大セール第二弾7/16〜7/31まで！

ツクモ 30TH YEAR 記念決算セール ザ・お得！

TSUKUMO SINCE 1962

毎年恒例！秋葉原電気まつり
賞金総額4000万円！
好評実施中7/26(日)まで
店頭にてお買い上げ￥5,000毎に抽選券1枚差し上げます。

シャープX68000の事なら何でも揃う！
ツクモにおまかせ！

歩き回る必要はありません！
まっすぐツクモへ

情報が沢山。分らない事何でもお尋ね下さい。
目に優しい10.4型カラー液晶ディスプレイ（LC-10C1）も取り扱い中！
詳しくはお問い合わせ下さい。
システムのご相談は☎03(3251)1899までどうぞ！

シャープよりいろんなパソコン！どれを選ぶ？

ビジネスでバリバリ持ち歩く方へ

DOS/V対応のOADG仕様SLノート
PC-6700シリーズがお勧め！

●32ビットCPU386SL（20MHz）を搭載。●パワーマネジメント機能によりバッテリ駆動時間の延長。●レターサイズのコンパクトなボディ（重さ2.2kg）に、ハードディスクとフロッピーディスクを内蔵。●高速液晶ディスプレイを採用。また、マイクロトラックボールを内蔵。

PC-6781J 定価￥630,000
3.5″1.44MB FDD1基・80MB HDD内蔵

決算特価販売中！

ホビーでガンガン使いこなす方へ

X68000 Compact XVI

●X68000の未来を象徴するハイコンパクトなボディ（体積比44%）●成熟するウィンドウ環境、使い易さと高機能を追求したSX-WINDOW Ver2.0搭載●2HD 3.5インチフロッピーディスクドライブ2基搭載●カラー液晶ディスプレイ接続可能●X68000XVIの高性能を継承●VGAモードサポート（SW-WINDOWのみ対応）

ワープロユースが中心で更にDOS/Vマシンのソフトを使う方へ

「書院パソコン」がお勧め **PC-WD1シリーズ** 定価￥330,000より

●スーパーアウトラインによる美しい印刷、すぐれた日本語処理能力。●ワープロ「書院」の先進機能をそのまま継承。●ハードディスク内蔵（Dタイプ）、OADG仕様、DOS/V対応。●CPUは32ビット80386SXを搭載。

決算特価販売中！

激安！

ツクモお勧めCompactセット

●CZ-674C-H（X68000 Compact本体）	￥298,000
●CZ-608D-H（0.28mmピッチCRT）	￥94,800
●100MBハードディスク	￥128,000
合計定価	￥520,800

決算特価販売中！

ツクモお勧めXVIセット

●CZ-634C-TN（本体）	￥368,000
●CZ-606D-TN（モニタ）	￥79,800
●100MBハードディスク	￥128,000
合計定価	￥575,800

決算特価販売中！

TSドライブ（X68000用）

目のつけどころがツクモでしょ。

X68000シリーズ専用3.5インチフロッピーディスクドライブ

TS-3XRシリーズ

〈仕様〉●3.5インチ2DD/2HD/2HCフォーマット対応の為、いろいろなフォーマットのメディアを読み書きができます。●ユーティリティソフト付属（デバイスドライバー/フォーマッター）
※初代X68KはROM交換が必要です。

3.5インチ1ドライブ **TS-3XR1** 定価￥44,800
特価￥35,800（消費税別途￥1,074）

3.5インチ2ドライブ **TS-3XR2** 定価￥57,800
特価￥46,800（消費税別途￥1,404）

※只今開発中。X68000CompactXVI用外付け5インチFDD

メモリー（X68000用）

■1MB増設RAMボード（CZ-600C専用） 特価￥19,500
■1MB増設RAMボード（ACE/PRO/PRO2シリーズ用） 特価￥17,000
■2MB増設RAMボード（拡張スロット用） 特価￥33,800
■4MB増設RAMボード（拡張スロット用） 特価￥59,800

※計測技研のメモリーボードも取り扱っております。価格についてはお尋ね下さい。

MIDIコンピュータミュージック（X68000用）

NEW Aセット

●CM-32L	￥69,000
●SX-68M-II	￥19,800
●Musicstudio Mu-1 Ver1.4	￥19,800
合計定価	￥108,600

特価￥88,000（消費税別途￥2,640）
クレジット例（18回払・税込）
初回￥5,863＋月々￥5,600×17回

NEW Bセット

●CM-300	￥58,000
●SX-68M-II	￥19,800
●Mu-1 SUPER	￥39,800
合計定価	￥117,600

特価￥92,000（消費税別途￥2,760）
クレジット例（10回払・税込）
初回￥10,919＋月々￥10,000×9回

NEW Cセット

●CM-500	￥115,000
●SX-68M-II	￥19,800
●Mu-1 SUPER	￥39,800
合計定価	￥174,600

特価￥141,000（消費税別途￥4,230）
クレジット例（15回払・税込）
初回￥11,300＋月々￥10,500×14回

SUPERマニアセット

●CM-64	￥129,000
●SX-68M-II	￥19,800
●Mu-1 SUPER	￥39,800
合計定価	￥188,600

特価￥154,000（消費税別途￥4,620）
クレジット例（18回払・税込）
初回￥10,261＋月々￥9,800×17回

※この他の組み合わせは、お問い合わせ下さい。☎03-3251-9911へ
※Mu-1 Ver1.4は3.5インチのメディアはありませんのでご注意下さい。

ローランド 追加オプション機器
ステレオマイクロモニター CS-10 定価￥~~17,000~~
MIDIキーボードコントローラー PC-200 定価￥~~36,000~~

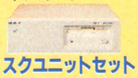
電子文具

シャープ 電子マネジメント手帳
PV-F1 定価￥128,000

●タイムマネージメントを管理する便利ツール新登場●従来の電子システム手帳用ICカードがそのまま使えます●次から次へと忙しい方への為の強力な助っ人●大画面・大容量・手書き入力で操作効率向上。

決算特価販売中！

ツクモはSONY MOディスクの正規販売代理店です



これが今一番の人気者！
SONY 3.5インチ光磁気ディスクユニットセット

●RMO-S350（3.5′光磁気ディスクドライブ）	￥235,000
●SCSIケーブル	￥6,900
●SCSIインターフェースボード	￥29,800
合計定価	￥271,700

シャープ純正「CZ-6MO1」も特価販売中！
決算特価販売中！

X68000用 SCSI HDD

200MB 決算特価 ￥110,000
100MB 決算特価 ￥76,000

画像処理

INPUT
■JX-220X A4サイズカラーイメージスキャナー……定価￥~~168,000~~

OUTPUT
■IO-735X-B カラーイメージジェットプリンター……定価￥~~248,000~~
■BJC-820J カラーバブルジェットプリンター……定価￥~~398,000~~
■CZ-8PC5-BK カラー熱転写プリンター……定価￥~~96,800~~

パソコン通信

時代は9600ボーへ!!
■モデム 9600bps/MNP5&CCITT V.42bis……特価￥69,800
■通信ソフト た〜みのる2……特価￥14,000

ソフトの事ならこの2店

ソフト8号店
ゲームソフト、ファミコンソフトなんでもかんでもとにかく安くて品揃え豊富
☎03(3251)0099
営AM10:15〜PM7:00

パソコン本店2F
X68000のソフトなら何でも揃っています。
☎03(3253)1899
営AM10:15〜PM7:00

ツクモソフト8号店B1 安心 迅速 高額 買い取りのツクモ買取センター

定休日	営業時間
毎週水・木曜日	AM11:00〜PM6:00

TEL. 03(3251)9977
FAX. 03(3251)5799

ツクモグローバルカード

大人気！入会者募集中！

18才以上なら学生さんもOK！

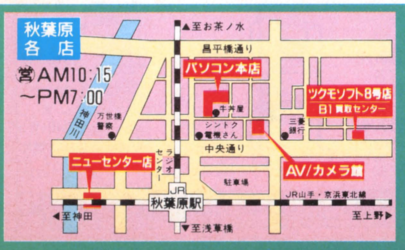
国内・外で活躍！使って便利、持ってて安心！ツクモグローバルカードはジャックス・VISAとの提携カードです。ツクモ各店でのお買物がらくらくできる上に、国内はもとより海外での分割ショッピングもOK！
20才以上の方にはキャッシングカードも発行致します。
お申し込みは☎03(3251)9898又は店頭にて！

全国どこからでも通話料無料

通信販売のご注文は下記フリーダイヤルへ。
受・注・専・用フリーダイヤル 0120-377-999
通販センター 03-3251-9911
商品についてのお問い合わせは各店又は通販へ。

クレジット払い 月々￥3,000以上の均等払いも頭金なし、夏・冬ボーナス2回払いも受付中！	**各種リース払い** くわしくは各店にお問い合わせ下さい。ケースに合わせてご相談にのります！	**現金書留払い** 〒101-91 東京都千代田区神田郵便局私書箱135号 ツクモ通販センター Oh!X係
カード払い 通信販売での御利用カード、ツクモグローバルカード、VIPカード、セントラル、ジャックス※御本人様より電話で通販部へお申し込み下さい。	**全国代金引き換え配達** お申し込みは☎03-3251-9911へお電話1本！ 配達日の指定もできます。	**銀行振込払い** 事前に☎でお届け先をご連絡下さい。 三和銀行 秋葉原支店（普）1009939 ツクモデンキ

超低金利冬・夏ボーナス二回払い受付中！お問い合せください。



ツクモは「スーパーX PRO SHOP」です。

PRO STAFF ツクモ

パソコン本店/荒井

九十九電機㈱
〒101-91 東京都千代田区神田郵便局私書箱135号
★商品のご注文は在庫確認の上お願いします。★表示価格には消費税は含まれておりません。

7月中は全店無休

ツクモパソコン本店2F ☎03-3253-1899（直通）
パソコン本店代表 ☎03-3253-5599（担当／荒井） ㊡毎週木曜日と8/12

ツクモニューセンター店	☎03(3251)0987	㊡毎週木曜日と8/12
名古屋1号店	☎052(263)1655	㊡毎週火曜日と8/17〜19(8/11は営業)
名古屋2号店	☎052(251)3399	㊡毎週水曜日と8/19〜21(8/12は営業)
ツクモ札幌店	☎011(241)2299	㊡毎週木曜日と8/5
DEPOツクモ2番街店	☎011(242)3199	㊡毎週木曜日と8/5

※但し7月中は全店無休

P&Aならではの新品パソコン

5年保証

《業界No.1の"P&Aメンテナンスサポート》

最高の保証システム

①業界最長の新品パソコン5年保証
（※モニター・プリンター３年間保証.// ※一部商品は除きます。）
②中古パソコンの1年間保証
（モニター・プリンター６ヶ月間保証）
③初期不良交換期間3ヶ月
（※新品商品に限らせていただきます）
④永久買取保証
⑤配達の指定OK//
⑥夜間配送もOK//
（※PM6:00〜PM8:00の間 ※一部地域は除きます。）

便利でお得な支払いシステム

①翌月一括払い手数料無料（ご利用下さい。）
②業界No.1の低金利
③月々の支払いは¥1,000より
④9ヶ月先からのスキップ払いOK//
⑤84回までの分割、ボーナス併用OK//
⑥カレッジクレジット
⑦ステップアップクレジット
⑧ボーナスだけで10回払いOK//
⑨現金一括払いOK//
（※商品・金額ご確認の上、銀行振込・現金書留にてご入金下さい。）

またまた

7/18〜8/18

《増設メモリー&数値演算プロセッサ》計測技研

1 PRKII-02(2M)……定価¥ 55,000▶特価¥ 39,800
2 PRKII-04(4M)……定価¥ 90,000▶特価¥ 67,000
3 PRKII-06(6M)……定価¥125,000▶特価¥ 92,500
4 PRKII-08(8M)……定価¥160,000▶特価¥119,000
5 PRKII-12(2M)……定価¥ 85,000▶特価¥ 63,000
6 PRKII-14(4M)……定価¥120,000▶特価¥ 89,500
7 PRKII-16(6M)……定価¥155,000▶特価¥114,500
8 PRKII-18(8M)……定価¥190,000▶特価¥141,000
9 MC-68881RC……定価¥ 38,000▶特価¥ 27,000

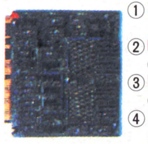
カラーイメージジェット
■IO-735X-B
定価¥248,000
特価¥152,000
（送料・消費税込み¥157,590）

■Z's STAFF PRO 68K Ver.3.0
（ツアイト）（定価¥58,000）
特価¥36,500
（送料・消費税込み¥38,110）

■SX-68MII（MIDI）
（サコム）定価¥19,800
特価¥13,500
（送料・消費税込み¥14,420）

■HGS-68（スキャナ）
（HAL研）定価¥39,800
特価¥24,500
（送料・消費税¥26,265）

X68000メモリボード（I/O・DATA）（送料¥500）



① SH-6BE1-1M（600CE用）定価¥25,000
（送料・消費税込み¥18,952）・特価¥17,900
② PIO-6BE1-A 定価¥25,000
（送料・消費税込み¥16,583）‥特価¥15,600
③ PIO-6BE2-2M 定価¥50,000
（送料・消費税込み¥32,239）‥特価¥30,800
④ PIO-6BE4-4M 定価¥88,000
（送料・消費税込み¥55,620）‥特価¥53,500

FDD(5インチ×2基)
■CZ-6FD5
（シャープ）（定価¥99,800）
P&A超特価!!
TEL下さい。

注目!!翌月一括払い手数料(金利)無料
（7月末/8月末/9月末のいずれかをご指定下さい）

X68000 CompactXVI/XVI/XVI-HD

※送料¥2,000、消費税別

今月の特選!! 特価品

■Compact XVI

● CZ-674C-H
● CZ-608D-H
● CZ-6FD5（5″FDD）
定価¥492,600
P&A超特価¥320,000
（※X68000サービスゲーム全て付いています。）
（モニターをCZ-606Dに変更の場合¥10,000を引いて下さい）

右記セットでお買い上げの方にもれなくプレゼント!!
①「ダウンタウン熱血物語(¥8,800)」
はもちろん、さらにその上、人気の
㋑「ロードス島戦記(¥9,800)」
㋺「グラディウスII(¥9,800)」
㋩「ザ・プロサッカー68(¥9,800)」
㋥「信長の野望武将風雲録(¥9,800)」
㋭「ELLE(エル)(¥7,800)」
の中のいずれか2本をプレゼント!!

X68000-CompactXVI ▶セットでお買い上げの方に●ディスケット10枚●ジョイカード2ケプレゼント中!!

Ⓐセット：CZ-674C＋CZ-608D……定価¥392,800▶特価¥281,000

12回	24,700	24回	13,000	36回	9,000	48回	7,000	60回	5,900

X68000-XVI ▶セットでお買い上げの方に●ディスケット10枚●ジョイカード2ケプレゼント中!!

Ⓐセット：CZ-634C-TN＋CZ-606D-TN……定価¥447,800▶特価価格はTEL下さい。

12回	26,200	24回	13,800	36回	9,600	48回	7,500	60回	6,300

Ⓑセット：CZ-634C-TN＋CZ-614D-TN……定価¥503,000▶特価価格はTEL下さい。

12回	29,700	24回	15,700	36回	10,800	48回	8,200	60回	7,100

X68000-XVI-HD ▶セットでお買い上げの方に●ディスケット10枚●ジョイカード2ケプレゼント中!!

Ⓐセット：CZ-644C-TN＋CZ-606D-TN……定価¥597,800▶特価価格はTEL下さい。

12回	35,700	24回	18,800	36回	13,000	48回	10,200	60回	8,600

Ⓑセット：CZ-644C-TN＋CZ-614D-TN……定価¥653,000▶特価価格はTEL下さい。

12回	39,000	24回	20,600	36回	14,300	48回	11,200	60回	9,400

※上記のモニターを、CZ-606D（定価¥79,800）、CZ-604D（定価¥94,800）、CZ-607D（定価¥99,800）、CZ-605D（定価¥115,000）、CZ-608D（定価¥94,800）、CZ-614D（定価¥135,000）、CU-21HD（定価¥148,000）に変更の場合、TEL下さい。超特価で販売致します。

X68000シリーズ〜P&Aスペシャルセット

（送料¥2,000・消費税別）

注目 スペシャルプレゼント!!
★ SUPER-HD には、
上記の①をプレゼント
★ PRO-II には、上記の
①＋㋑〜㋭の中の2本をプレゼント

ズバリ価格で大奉仕中

●ディスケット10枚、●ジョイカード2個プレゼント中

SUPER-HD P&A特選セット 限定

Ⓐセット：■CZ-623C-TN（単品）…………定価¥498,000▶特価¥218,000
Ⓑセット：■CZ-623C-TN＋CZ-606D……定価¥577,800▶特価¥272,000
Ⓒセット：■CZ-623C-TN＋CZ-608D……定価¥592,800▶特価¥284,000
Ⓓセット：■CZ-623C-TN＋CZ-607D……定価¥597,800▶特価¥286,000
Ⓔセット：■CZ-623C-TN＋CZ-614D……定価¥633,000▶特価¥306,000
Ⓕセット：■CZ-623C-TN＋CU-21HD……定価¥646,000▶特価¥316,000

PRO-II P&A特選セット 限定

Ⓐセット：■CZ-653C（単品）……………定価¥285,000▶特価¥138,000
Ⓑセット：■CZ-653C＋CZ-606D……………定価¥364,800▶特価¥195,000
Ⓒセット：■CZ-653C＋CZ-604D……………定価¥379,800▶特価¥197,000
Ⓓセット：■CZ-653C＋CZ-608D……………定価¥379,800▶特価¥207,000
Ⓔセット：■CZ-653C＋CZ-607D……………定価¥384,800▶特価¥209,000
Ⓕセット：■CZ-653C＋CZ-614D……………定価¥420,000▶特価¥229,000
Ⓖセット：■CZ-653C＋CU-21HD……………定価¥433,000▶特価¥239,000

X68000用ハードディスク

■HD-J040（システムサコム）（¥89,000）………
（送料・消費税込み¥64,890）‥‥▶特価¥ 62,000
■HD-J100（システムサコム）（¥128,000）………
（送料・消費税込み¥89,610）‥‥▶特価¥ 86,000
■HD-J130（システムサコム）（¥148,000）………
（送料・消費税込み¥106,090）‥▶特価¥102,000
■HD-J170（システムサコム）（¥189,000）………
（送料・消費税込み¥128,750）‥▶特価¥124,000

プリンター（ケーブル用紙付）

（送料¥1,000 消費税別）

■CZ-8PC5-BK 定価¥ 96,800▶特価¥69,000
■CZ-8PK10‥‥定価¥ 97,800▶特価¥71,000
■CZ-8PG2‥‥定価¥160,000▶特価価格はTEL
■CZ-8PG1‥‥定価¥130,000▶特価価格はTEL

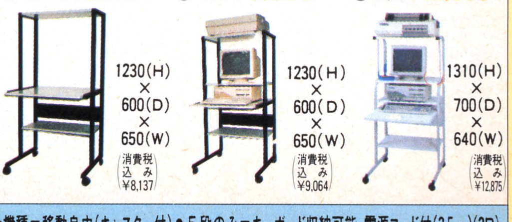
モデム

■PV-M24B5
（AIWA）（定価¥39,800）
▶特価¥25,000
（送料・消費税込み¥26,780）
■MD-24FB5V
（オムロン）（定価¥39,800）
▶特価¥25,500
（送料・消費税込み¥27,295）
■FMMD-311G
（富士通）（定価¥35,800）
▶特価¥24,800
（送料・消費税込み¥26,574）

P&A特選パソコンラック

（消費税別）（送料無料）

①3段¥7,900　②4段¥8,800　③5段¥12,500



① 1230(H)×600(D)×650(W)（消費税込み¥8,137）
② 1230(H)×600(D)×650(W)（消費税込み¥9,064）
③ 1310(H)×700(D)×640(W)（消費税込み¥12,875）

全機種＝移動自由（キャスター付）●5段のみ＝キーボード収納可能、電源コード付(2.5m)(2P)

●本広告の掲載の商品の価格については、消費税は含まれておりません。　●営業時間＝平日AM10:00〜PM7:00、日祭AM10:00〜PM6:00

アフターサービス万全
全商品保証付。専門の担当がお客様の立場で対応します。
初期不良、輸送トラブルetc.
万が一初期不良、輸送トラブルが発生しました際には、即交換させていただきます。

★頭金なし!!
★即日発送!!

秋葉原でおなじみのP&Aがズバリ超特価セールでご奉仕!!

- ●お近くの方は、お立寄下さい。専門係員が説明いたします。
- ●本体単品でも受付します。詳しくは、お電話にてお問合せ下さい。
- ●ビジネスソフト定価の15%引きOK!! TEL下さい。
- ●現金書留及び銀行振込でお申し込みの方は、上記商品の料金に3%加算の上でお申し込み下さい。詳しくは、お電話にてお問い合せ下さい。

全国通販

X68000用ソフトコーナー （送料1ヶ〜5ヶまで¥500・消費税別）

◆Z's STAFF PRO68K Ver3.0(ツアイト)	定価¥58,000▶	特価¥36,500
◆Z's TRIPHONY デジタルクラフト(ツアイト)	定価¥39,800▶	特価¥27,000
◆テラッツォ(ハミングバード)	定価¥19,400▶	特価¥13,600
◆マジックパレット(ミュージカルプラン)	定価¥19,800▶	特価¥14,200
◆Gツール(ザインソフト)	定価¥28,000▶	特価¥18,600
◆たーみのる2(SPS)	定価¥17,800▶	特価¥13,000
◆Mu-1 Super	定価¥39,800▶	特価¥28,500
◆サイクロンEXPRESSα68	定価¥98,000▶	特価¥69,000
◆KAMIKAZE(サムシンググッド)	定価¥68,000▶	特価¥43,800
◆C-TRACE68 Ver3.0(キャスト)	定価¥98,000▶	特価¥68,500
◆G68K Ver.2 PRO	定価¥22,000▶	特価¥17,300
◆C&Professional Pack(マイクロウェアジャパン)	定価¥58,000▶	特価¥39,800
◆Final Ver3.2(エーエスピー)	定価¥38,000▶	特価¥29,000
◆CZ-213MSD MUSIC PRO68K	定価¥18,800▶	特価¥13,200
◆CZ-214MSD SOUND PRO68K	定価¥15,800▶	特価¥11,300
◆CZ-215MSD Sampling PRO68K	定価¥17,800▶	特価¥12,500
◆CZ-220BSD DATA PRO68K	定価¥58,000▶	特価¥40,000
◆CZ-224LSD The 福袋 Ver2.0	定価¥ 9,900▶	特価¥ 7,400
◆CZ-225BSD Multiword Ver1.1	定価¥32,000▶	特価¥23,000
◆CZ-243BSD CYBERNOTE PRO68K	定価¥19,800▶	特価¥15,000
◆CZ-247MSD MUSIC PRO68K[MIDI]	定価¥28,800▶	特価¥20,500
◆CZ-249GSD CANVAS PRO68K	定価¥29,800▶	特価¥22,000
◆CZ-251BSD Hyper word	定価¥39,800▶	特価¥29,400
◆CZ-253BSD CARD PRO68K Ver2.0	定価¥29,800▶	特価¥22,700
◆CZ-257CSD Communication PRO68K Ver2.0	定価¥19,800▶	特価¥15,300
◆CZ-258BSD Teleportion PRO68K	定価¥22,800▶	特価¥16,900
◆CZ-261MSD MUSIC studio PRO68K Ver2.0	定価¥28,800▶	特価¥21,200
◆CZ-263GWD Easypaint SX-68K	定価¥12,800▶	特価¥ 9,800
◆CZ-265HSD New Print Shop Ver2.0	定価¥20,000▶	特価¥15,400
◆CZ-266BSD Press Conductor PRO68K	定価¥28,800▶	特価¥22,000
◆CZ-284SSD OS-9/X68000 Ver2.4	定価¥35,800▶	特価¥25,600
◆CZ-285LSD C-Compiler PRO68K Ver2.1	定価¥44,800▶	特価¥32,500
◆CZ-286BSD BUSINESS PRO68K Popular	定価¥28,000▶	特価¥20,500
◆CZ-287SS SX-WINDOW Ver2.0	定価¥12,800▶	特価¥ 9,800

☆ゲームソフト25%OFF OK!!（一部ソフト除く）

周辺機器コーナー （送料¥500・消費税別）

①CZ-8NS1	定価¥188,000▶	特価¥133,000
②CZ-6VT1	定価¥ 69,800▶	特価¥ 49,500
③CZ-6TU	定価¥ 33,100▶	特価¥ 23,900
④BF-68PRO	定価¥ 19,800▶	特価¥ 14,400
⑤CZ-8NM3	定価¥ 9,800▶	特価¥ 7,200
⑥CZ-8NT1	定価¥ 13,800▶	特価¥ 10,000
⑦CZ-6BE2A	定価¥ 59,800▶	特価¥ 42,800
⑧CZ-6BE2B	定価¥ 54,800▶	特価¥ 39,300
⑨CZ-6BE2D	定価¥ 54,800▶	特価¥ 39,300
⑩CZ-6BF1	定価¥ 49,800▶	特価¥ 35,800
⑪CZ-6BP1	定価¥ 79,800▶	特価¥ 57,000
⑫CZ-6BM1	定価¥ 26,800▶	特価¥ 19,300
⑬CZ-6EB1	定価¥ 88,000▶	特価¥ 63,000
⑭AN-S100	定価¥ 36,600▶	特価¥ 26,300
⑮CZ-6SD1	定価¥ 44,800▶	特価¥ 32,500
⑯CZ-6BN1	定価¥ 29,800▶	特価¥ 21,500
⑰CZ-6BV1	定価¥ 21,000▶	特価¥ 15,200
⑱CZ-6BC1	定価¥ 79,800▶	特価¥ 57,000
⑲CZ-6BG1	定価¥ 59,800▶	特価¥ 43,000
⑳CZ-6BU1	定価¥ 39,800▶	特価¥ 28,500
㉑CZ-6PV1	定価¥198,000▶	特価¥142,000
㉒CZ-6BS1	定価¥ 29,800▶	特価¥ 21,500
㉓CZ-8NJ2	定価¥ 23,800▶	特価¥ 17,500
㉔CZ-6BL2	定価¥298,000▶	特価¥214,000
㉕JX-100S	定価¥ 89,800▶	特価¥ 44,000
㉖JX-220X	定価¥168,000▶	特価¥121,000
㉗IO-735XB	定価¥248,000▶	特価¥154,000
㉘LC-10C1H	定価¥598,000▶	特価¥459,000
㉙CZ-6CS1(674C用)	定価¥ 12,000▶	特価¥ 8,900

中古・高価現金買取り/下取りOK!!

■まずはお電話下さい。
下取り専用 買取り電話 ▶ ☎03-3651-1884 FAX. 03-3651-0141
■下取り・買取りで、お急ぎの方は、直接当社に来店、または宅急便にてお送り下さい。

買取り価格…完動品・箱/マニュアル/付属品付の価格です。

- ●下取りの場合……価格は常に変動していますので査定額をお電話で確認して下さい。（差額は、P&A超低金利クレジットをご利用下さい。）
- ●買取りの場合……現品が着き次第、2日以内に買取り金額を連絡し、振込み、又は書留でお送り致します。
- ●近郊の方は、P&A本店まで、直接お持ち下さい。即金にて、¥1,000,000までお支払い致します。

- ●最新の在庫情報・価格はお電話にてお問い合せください。
- ●買い取りのみ、または、中古品どうしの交換も致します。詳しくは電話にて、お問い合せ下さい。
- ●価格は変動する場合もございますので、ご注文の際には必ず在庫をご確認下さい。
- ●本商品の掲載の価格については、消費税は、含まれておりません。
- ●現金書留及び銀行振込でお申し込みの方は、上記商品の料金に3%加算の上でお申し込み下さい。詳しくは、お電話でお問い合せ下さい。

《便利な超低金利クレジットをご利用下さい》

●月々¥1,000円からOK!! ●ボーナス払いOK（夏冬10回までOK）
●支払い回数 1回〜84回 ●お支払いは、8ヶ月先からでもOK!!

●定休日/毎週水曜日

P&A特選＝今月の中古特選品

●CZ-601C ●CZ-611D-TN ¥120,000	●CZ-634C-TN ●CZ-606D-TN ¥248,000	●CZ-644C-TN ●CZ-604D-TN ¥318,000

買取り価格

●CZ-634C	¥170,000	●CZ-602C	¥75,000
●CZ-644C	¥230,000	●CZ-612C	¥85,000
●CZ-604C	¥100,000	●CZ-652C	¥55,000
●CZ-623C	¥138,000	●CZ-662C	¥75,000
●CZ-603C	¥ 85,000	●CZ-611C	¥68,000
●CZ-613C	¥105,000	●CZ-601C	¥45,000
●CZ-653C	¥ 75,000	●CZ-600C	¥45,000
●CZ-663C	¥ 90,000		

下取り交換差額表

新品 / 下取り	CZ-634C モニターセット	CZ-644C モニターセット	モデル UX20セット	モデル CX20セット	9801DA2
CZ-623C モニターセット	150,000	270,000	70,000	160,000	130,000
CZ-613C モニターセット	190,000	290,000	100,000	190,000	160,000
CZ-652C モニターセット	230,000	340,000	150,000	240,000	220,000
CZ-604C モニターセット	180,000	290,000	100,000	190,000	160,000
CZ-600C モニターセット	230,000	340,000	150,000	240,000	220,000

通信販売お申し込みのご案内

〔現金一括でお申し込みの方〕
●商品名およびお客様の住所・氏名・電話番号をご記入の上、代金を当社まで、現金書留でお送りください。（プリンター・フロッピーの場合、本体使用機種名を明記のこと）
〔銀行振込でお申し込みの方〕
●銀行振込ご希望の方は必ずお振込みの前にお電話にてお客様のご住所・お名前・商品名等をお知らせください。
（電信扱いでお振込み下さい。）

〔振込先〕 住友銀行 新小岩支店
普通預金 1451576 ㈱ピー・アンド・エー

〔クレジットでお申し込みの方〕
●電話にてお申し込みください。クレジット申し込み用紙をお送りいたしますので、ご記入の上、当社までお送りください。
●現金特別価格でクレジットが利用できます。残金のみに金利がかかります。
●1回〜84回払いまで出来ます。但し、1回のお支払い額は¥1000円以上。

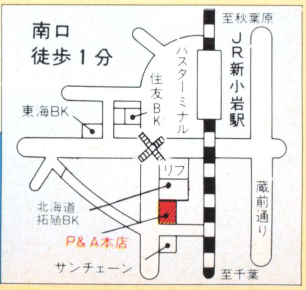
超低金利クレジット率

回数	3	6	10	12	15	24	36	48	60	72
手数料	3.0	4.0	5.5	5.5	8.5	11.5	16.0	21.0	27.0	33.0

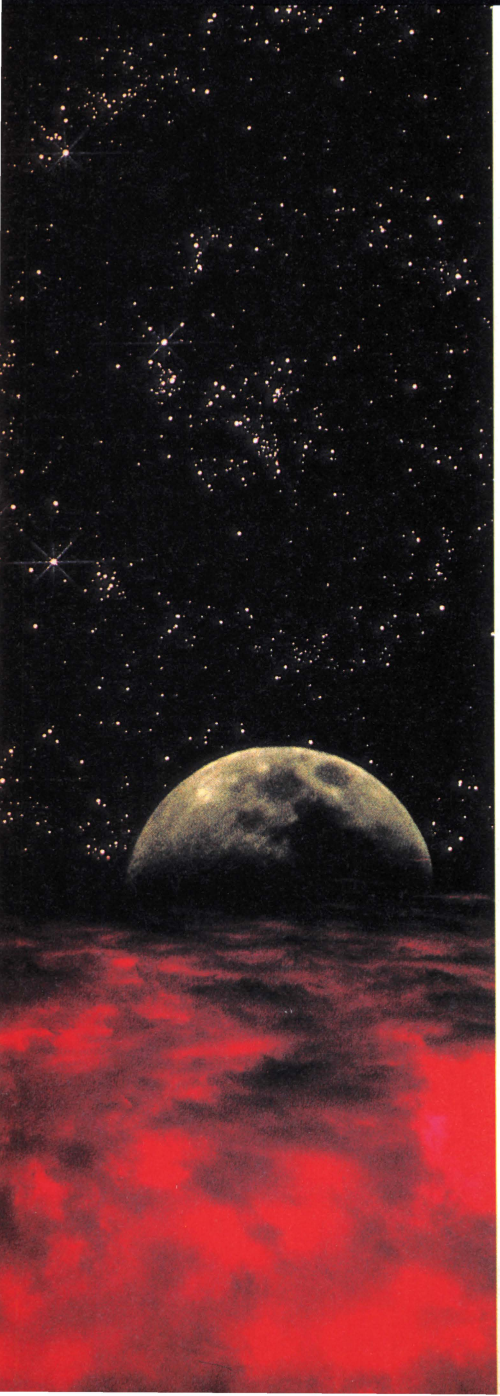
注目!!翌月一括払い手数料(金利)無料（7月末/8月末/9月末のいずれかをご指定下さい）

マイコン専門ショップ P&A
株式会社ピー・アンド・エー
〒124 東京都葛飾区新小岩2丁目1番地19号
☎03-3651-0148（代） FAX. 03-3651-0141

営業時間
平日：AM10:00〜PM7:00
日祭：AM10:00〜PM6:00



●現金書留及び銀行振込でお申し込みの方は、上記商品の料金に3%加算の上でお申し込み下さい。詳しくは、お電話でお問い合せ下さい。



アプリケーション指向のUNIX活用誌

月刊［ユニックス・ユーザー］

UNIX USER

毎月8日発売
A4変型判・本文144頁
定価980円（税込）

ユーザーの視点から、今日、将来のUNIXの世界を展望します。

創刊号特集

Solaris2.0の全貌

SunOSの新機軸とSPARC、i486によるマルチプラットフォーム戦略

Part1●Solaris2.0で変わるエンドユーザー環境

Part2●日米サン・ソフト代表インタビュー　エド・ザンダー ほか

Part3●Solarisを有力視する国内有力ソフトハウス
——ジャストシステム　浮川和宣

7月8日創刊

ビジネスUNIX講座

●マルチプラットフォーム・コネクション
マッキントッシュ用 TCP／IPの導入

●オフィス・エリア・ネットワーク
LaMailでUNIX、Windows、Macの電子メールを統合する

●パソコンから使うUNIXデータベース
InformixとWing Zによるクライアントサーバ環境

UNIX USER LAB

日本電気EWS4800のユーザー環境と開発環境

UNIX入門からシステム管理・拡張講座

UNIXのグラフィカル・ユーザ・インタフェース：X-Windowの導入
これで使える！ システム設計学：ハードディスクのパーティション設定
whatis UNIX：ログインとターミナルタイプの設定
インターネット構築術：シリアルポートからのログイン
実践UNIX：Cシェル基礎の基礎
デバイス活用術：キヤノンLASER SHOTをワークステーションで使う

付録ディスク　UNIX USER LIBRARY

SOFT BANK　ソフトバンク出版事業部
〒108 東京都港区高輪2-19-13 NS高輪ビル
TEL 03(5488)1360

月刊PC 10月創刊

モノが主役の、まったく新しいパソコン誌です。

「PC」では、投稿を募集します！

新パソコン情報誌PCは、パソコンユーザー参加型メディアです。
あなたの生の声を、広く全国のパソコンユーザーに伝えます。

実際にパソコン関連製品を使ってみて、肌で感じた製品の良かった点、ガッカリした点を書いて、他のパソコンユーザーに教えてあげてください。

秋葉原通信、ハードやソフトのオリジナル活用法、サポートの不平不満、長期稼働マシン自慢etc.…。さまざまなコーナーを設けて、あなたの原稿をお待ちしています。

破格の原稿料で、お応えします！

採用分の投稿には原稿料をお支払いいたします。投稿は原稿用紙でもフロッピー（DOSフォーマット）でもOK。ただし原稿は400字詰で7枚までとさせていただきます。

宛先は
〒108 東京都港区高輪2-19-13 NS高輪ビル
ソフトバンク㈱出版事業部 「月刊PC」投稿係

ソフトバンク株式会社
出版事業部 〒108 東京都港区高輪2-19-13 NS高輪ビル
TEL03-5488-1360 FAX03-5488-1364

［第15回］
羽

寺尾　響子

響子inCGわ〜るど

CGからはちょっと離れてしまうけれど，どうしてもしておきたい話があります。身近にいたひとのことです。そのひととはずいぶん長いあいだ，いろいろなことを一緒にしてきました。バンド活動もしたし，旅もしたし，買い物や料理もして，本当にたくさんの時間を過ごしてきたのです。

そのひとは，いわゆる少年の心をもった大人でした。

ある日，そのひとが沈んだ目をして話しかけてきました。

「背中にまだ羽が生えているかと思って，ずっと頑張ってきたけれど，やっぱり羽はもげていたよ……」

ポツリというのでした。そして，さらにこう付け加えました。

「また生えてくるかと待ってみたけれど，だめだった」と。

落ちた羽

最初は，「何をいっているんだろう，このひと」と思いました。そのうち様子が飲み込めてきて，頭がクラクラとしてきました。

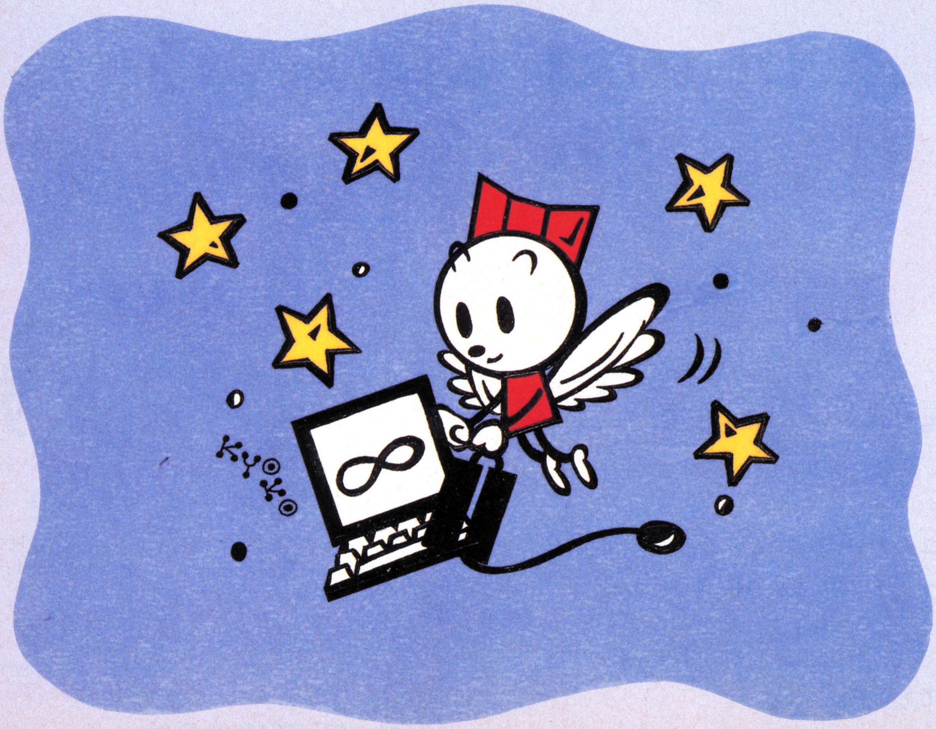
つまり，こういうことなのです。

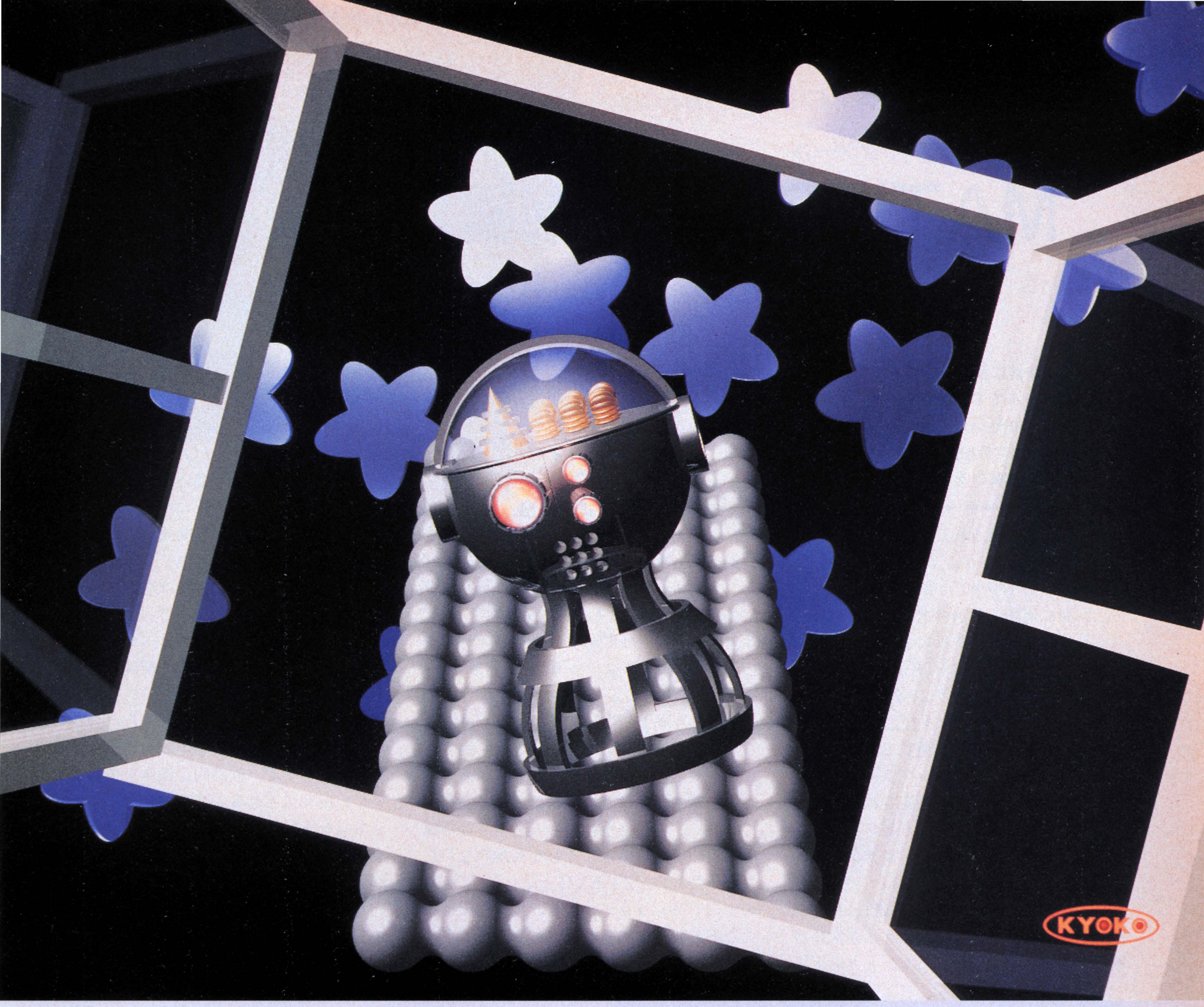
自分自身の中から湧き出てくるイメージをかたちにする仕事，ミュージシャンやゲームデザイナー，作家，絵描きなんかがやっているような仕事をしたかったのです。そのひとは，フォトグラファーになりたいと考えていました。

サラリーマン生活のかたわら，写真を撮りつづけました。どんどん上手になっていくのが作品を通じてわかります。ポスターになったり，雑誌に掲載されたりして，とても順調に見えました。

しかし，あるときこういったのです。

「ハネガ　モゲタ……」





もう夢見て飛ぶことはできない，と伝えたかったのです。手助けができなかった私自身が，情けなくて悲しくなりました。

もげて落ちたそのひとの白い羽が，床にふたつゴロンと転がっているように見える気がします。

背中に羽の根元のかたちをした生々しい傷痕が浮かんでくるようでした。ちょうど，走っていて転び，擦りむいて，乾ききらない血が残っている，ひざ小僧のけがみたいな。

＊　　＊　　＊

今日もコンピュータのウィンドウを開けます。自分の意識を重ね合わせ，するりと窓を抜けて，もうひとつの空間へ。

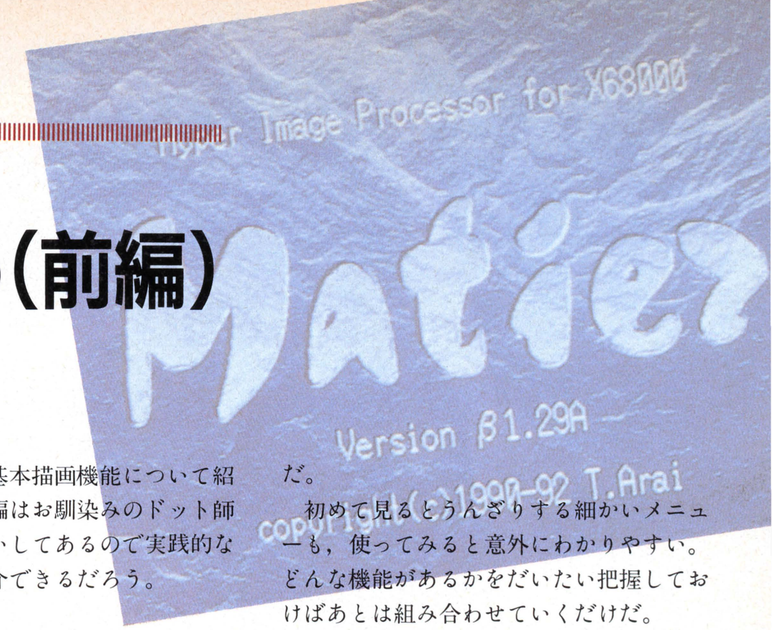
飛びながら，お願いをします。

「カミサマ　イツマデモ　トビツヅケテイラレマスヨウニ……」

もうひとつお願いをします。

いままでずっと大切だった，そして，これからも大切な友人であるひとのために。

「カミサマ　カレニ　フタタビ　ハネガ　ハエテキマスヨウニ……フタタビ　ハネガ　ハエテキマスヨウニ……」



多機能グラフィックツール

MATIERを使う(前編)

Nakano Shuichi 中野 修一

MATIERは非常に多彩な機能を持ったグラフィックツールです。独自のノウハウで特に3D CGとの相性は最高といえるでしょう。もちろん普通にCGを描く際にも，画像処理をする場合にも威力を発揮します。

寺尾響子さんの連載でも使用ツールとして挙げられていたり，7月号で紹介したNHKのCG制作現場でも使用されていたりと，誌面に何回か名前の挙げられていたわりに具体的な解説がされなかったグラフィックツール，それがMATIERだ。

実はこのツールは，もう2，3年前から開発が進められてきたツールだ。C-TRACEユーザーズクラブなどを通じてモニタリングを繰り返し，たび重なるブラッシュアップを経てついに市販化が決定された。発売はサンワード（と，聞いてもピンとこない人も多いと思うがサンワードはサン・ミュージカルサービスの新社名だ)。

MATIERは「マチエール」と読む。Hyper Image Processorと冠するように非常に多機能な画像加工ツールである。とりあえず今回は前編として基本描画機能について紹介しよう。なお後編はお馴染みのドット師川原由唯氏にお願いしてあるので実践的な内容は来月号で紹介できるだろう。

Beyond the Z's.

Z'sSTAFFはX68000で最初に発売されたグラフィックツールである。Z'sSTAFFはX68000でもっとも広く使われているグラフィックツールである。X68000ユーザーのグラフィックツールの基準は常にZ's STAFFであったといっていい。

しかしZ'sSTAFFが完璧だったわけではない。基本的には「あくまでも手で描く」古いかたちのグラフィックツールである。加えて基本機能は登場以来ほとんど変わっていない。もっと手軽にもっと多彩な処理をしたいという欲求。それを具体的なかたちにしたものがZ's-EXだった。そして，この春のバージョンアップで，拡張性を持ったZ'sSTAFFは磨きあげられて実に「よいツール」になった，と思う。

なぜこんなことを書くかというと，多くのグラフィックツールが挑んでも揺がなかったZ'sの牙城を，脅かしそうなツールがようやく現れたからだ。MATIERは画像を「処理し加工する」という点で，Z's STAFFとは性格を異にした強力なグラフィックエディタである。

両者の最大の違いはユーザーインタフェイス。どちらがいいともいえないが，マルチウィンドウで処理するZ's STAFFとアイコンを詰め込んですべての機能を一覧させるMATIER。これは好みの問題だ。

初めて見るとうんざりする細かいメニューも，使ってみると意外にわかりやすい。どんな機能があるかをだいたい把握しておけばあとは組み合わせていくだけだ。

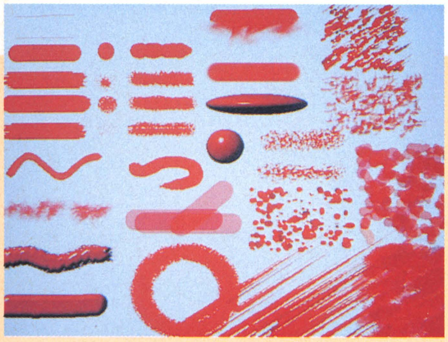
このメニューは描画時には表示されず，全画面がエディット用に開放される。右クリックでメニューを呼び出し，左クリックで選択する。場所によっては左右クリックでアップダウン，メニューを右クリックするとさらに詳細なサブメニューが現れる。基本的にメニュー階層は浅く配置されているので迷うことはないだろう。あっ，と，描画時以外の右クリックはもちろんスポイトである。こうして列挙するとあまり体系だっていないが，直感的な操作をすればたいてい応えてくれるシステムになっている。どちらかといえば，多彩なモードを使いこなすことのほうが難しいといえる。

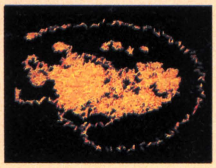
MATIERの特徴

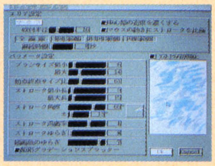
まず，スペックを並べよう。通常は512×512モードで動作するグラフィックエディタだ。最大32768色を使って，メモリの許す限りの大きな絵が描けるという特徴を持つ。

メモリは最低限2Mバイト必要。4M以上であれば使用できる裏画面の数や子プロセス用メモリなどで余裕を持った使い方が可能となる。裏画面はZ's-EXでいう裏画面とほぼ同じだが，大きさが画面以上の場合があることと最大4枚の裏画面が使えることが違いといえる。このほかUNDOバッファも効果的に使用できるので，MATIERが抱えることのできるデータは膨大であることがわかる。

ファイル形式はPICがほぼ標準で，そのほかにGL3を拡張したGLX形式，RGB，IMGなどのレイトレーシングソフトウェアで一般的に使用されている24ビットフルカラー形式，Macintosh式TIF，TOWNS式のTIFデータ（非圧縮）に対応している。あとは圧縮TIFとJPEGに対応すればほぼ完璧といっていいだろう。







右はMATIERのメインメニュー。右クリックを押すと画面に出現する。上段は色の設定，中段からは基本的な筆の設定，ツール類，エフェクト，特殊機能といった順にアイコン群が集められている。上は基本描画のいろいろ。ブラシは特に多機能だ。

対応するスキャナはエプソンGT-4000/6000とシャープCZ-8NS1（JX-220）のみとなっている。このあたりのサポートはZ'sSTAFFのほうに一日の長があるようだ。操作性はまずまずでプリスキャンによる範囲指定やミリ単位での大きさ指定などの配慮がうれしい。シャープ純正アプリとは違ってスキャナ用パラレルボードにも対応しているので高速な取り込みが可能だ。

プリンタはIO-735をはじめ，CZ-8PCシリーズやPC-PRシリーズのカラープリンタとビデオプリンタに対応している。モノクロプリンタがサポートされていないのが特徴的といえるだろう。

ファイル形式のところでも述べたようにMATIERは1600万色データを読み込み，エディットできる。これではせっかくの24ビットフルカラーデータが15ビットデータになってしまうのでもったいない。MATIERではエディットした部分のみを上書きする機能があるので，クオリティの低下は最低限ですむ。さらに画面モードを変えた正方形ドットモードもあるのでレイトレ屋さんの必需品といえるだろう。

とりあえず，Z'sSTAFFにできてMATIERでできない機能はタイル&スクリーントーンくらいのものだが，MATIERでできてZ'sSTAFFでできない機能は山のようにあるのは事実である。

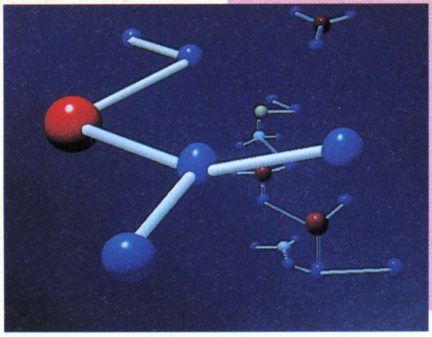
Z'sSTAFFのような拡張性はないが，画面を表示したまま子プロセスを起動できるのでフィルタプログラムは簡単に作成できる。問題は画面範囲を超える絵に対するフィルタリングだ。なんらかのインタフェイスがほしかったところだ。

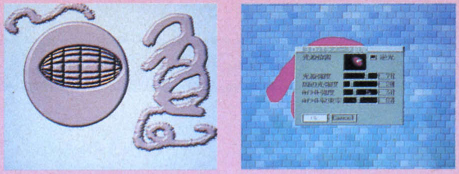
基本描画のバリエーション

Z'sSTAFFではペン先形状を自由に再定義できた。が，MATIERでは基本のペン先は数種類しかない。しかし，大きさが可変，透明度が可変，中心部の密度を濃くするかどうかの指定などが設定できる。さらに，描画時に線の太さを不均一にする指定やブラシの機能によって驚くほど多彩な表現力を発揮する。

写真を見てほしい。赤一色でペン先（の一部）を使ってみた。

最初は単なるライン，続いて基本のペン先とそれを均等線，不均等線で描画したものだ。不均等線のタッチがわかるだろうか。自動的に線の太さが変化する毛筆モードもある（ただし，制御は不可能に思えるほど難しいが）。これらのペン先は透明度を自由





3D関係の描画機能で描いたもの。レイトレ風のCGもひょいひょいと作れる。さらに3Dペイントを多用するとメタボールも目じゃない？　3Dパラメータ設定では，光源と周辺光ハイライトを決定する。

に変え，ボックスやサークル，ブラシなどでも使用できる。

そう，ブラシ。MATIERの基本描画機能でもっとも特徴的なものがこのブラシだ（写真の右半分）。これは先ほどのペン先を使って1回のマウス操作でたくさんの描画を行うもの，と思えばいい。たとえば，描画範囲を矩形にしておくと指定された範囲内にランダムかつ均等にペンの描画が行われる。Z'sSTAFFではまさにエアブラシのようにドットしか使えなかったが，MATIERではドットだけでなくペン先のパターンが使える。これが基本形だ。

さらに，使用するペン先の大きさを，指定した範囲でランダムに変化できる。

さらに，単にペン先を置くだけでなく，ペンにタッチを加えることができる。具体的にいうと，そのペン先を使った短いラインを使ってブラシを行う。その際のストローク長は指定した範囲でランダムに変化する。もちろん線の角度は自由に設定できる。

さらに，そのストロークの最初と終わりで線の幅を変えることができる。

さらにストローク角度に幅を持たせて範囲内でランダムに指定できる。

さらに，その線に曲率を指定できる。

さらに，基本直線または曲線にゆらぎを与えることができる。これを使うと少し震えたような手描きっぽい線になる。

さらに，描画色にゆらぎを指定できる。

とまあ，さまざまな指定ができるわけだ。このように多機能なブラシはいったいなんのためにつけられたのかというと，油絵などで見られる人間のタッチを再現するためにほかならない。もっとも，パラメータの指定範囲が広いのでもっと特殊な効果にも使用できるのはいうまでもない。

当然，透明度や不均一線の指定は併用可能だし，このブラシ自体でエフェクト領域を指定する（というか，リアルタイムにエフェクトを加える）こともできる。このあたりの，いわば直交性のよさはMATIERの機能全般に通じた美点である。

ブラシ以外のペンとして，もやもやしたフラクタルペンや，水彩画で水をつけて擦るような効果を出すペンもある。パーソナルの閾を超えて，PC-9801のフルカラーグラフィックエディタSuperTableauに迫る多彩な機能を誇っている。

その他の描画機能

ボックスやサークル，ペイントなどは既存のツールと大差ないので省略。

球を描く機能がある。文字どおり，指定した色で円ではなく球を描く。疑似的にグラデーションを加えたものではなく，光源を設定し，アトリビュートを指定してフォンシェーディングで球体を描画する。ほかになにができるわけではないが，手描きだと難しいピカピカの球やプラスティック風の球を簡単に描き分けることができる。

このように，MATIERは2Dグラフィックツールでありながら，3Dグラフィック特有の機能をいくつか取り込んでいる。圧巻は3Dペイントとでも呼ぶべきもっこりペイントだ。写真にある「なんとなくもっこりしたもの」は，描画した線に3Dペイント処理を加えたものだ。これはペイントする領域の境界線から疑似的に法線を算出してフォンシェーディングにかけるもので，見てのとおりの効果を発揮する。別にもっこりでなく，えぐれた形にすることもできるし光源設定などもできる。

以上，MATIERの特徴を基本機能レベルで簡単に紹介してみた。これでは重いのではないかと心配する人もいるだろうが，そんなことはないので安心してほしい（重いことをやらせると重いが）。

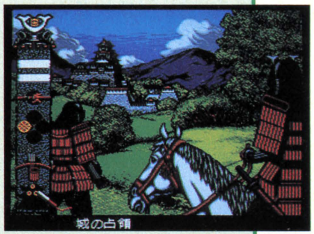
MATIERは基本ツール部分だけ見ても十分に魅力的なツールである。ブラシや3Dペイントはあっというまに画面に質感を醸しだす。絵描きでない私にでも「絵らしきものを描かせてくれる」ツールなのだ。

しかし，最初に述べたように「画像処理ソフトウェア」MATIERの真骨頂は特殊効果やエディット機能にある。来月のレポート後編をお楽しみに。

MATIER　39,800円
サンワード　☎044(853)2580

SOFTWARE INFORMATION

7月17日に「ファイナルファイト」が発売された（はずだ）けど，いち早く手に入れて，遊びまくっている人もたくさんいるだろうなあ。来月号ではみっちりと紹介したいと思うのでお楽しみに。



ライジングサン

海の向こうの“ニッポン”ゲーム，つまり日本を題材とした海外ゲームというと，どうしてもウサン臭いという偏見をもってしまう。しかし，この「ライジングサン」は日本でアレンジされたせいか，もとからそうなのか，わりとマトモな雰囲気が漂っている。

もともとは海外のシネマウェアという，ビジュアルやドラマ性を重視したシリーズのひとつとして発売されたもの。源氏一族の大将（頼朝か，義経かを選べる）になって，平家を倒し，全国統一を目指さねばならない。ゲームは常にリアルタイムで進められ，合戦や城攻めなどはアクションゲーム仕立てになっている。

戦略画面と呼ばれる全体マップではあちこちで武将が歩き回り，空には雲が流れ，どこそこで戦闘が起きた，誰々が負けたという表示がされる。シミュレーションゲームにありがちなオカタい空気のカケラもないのが最大の特徴だ。

X68000用　5″2HD版　9,800円（税別）
ビクター音楽産業　☎03(3423)7901

もうすぐ夏だね，新作だね

1.	グラディウスII	1
2.	ファイナルファイト	3↑
3.	ジェノサイド2	4↑
4.	スターウォーズ	2↓
5.	三國志III	ー初
6.	レミングス	7↑
7.	シムアース	ー初
8.	ライフ＆デス	6↓
9.	OVERTAKE	ー初
10.	出たな!! ツインビー	5↓
	ポピュラスII	ー初

毎月，アンケートハガキの「推薦する市販ソフト」欄からゲームソフトを抽出，集計してお届けする“今月のTOP10”。6月号のハガキの集計結果を発表いたしましょう。

またまた「グラディウスII」が1位となりました。王者の座についてもうかれこれ5カ月になりますが，すぐ背後には「ファイナルファイト」が肉薄。そろそろ世代交代の足音が聞こえてきます。

超絶移植の威光でトップの座を守れるのか，「グラディウスII」。アーケードゲーム「ストリートファイターII」のブームを追い風に首位を奪えるのか，「ファイナルファイト」。来月あたりがヤマになりそうな予感。

5位は今月レビューされている「三國志III」ですが，ユーザーの評判は上々のよう。前作より難易度は上がったようですが，コマンドの充実やオリジナル武将が作れるなど，やはりシリーズNo.1の完成度は間違いないようです。唯一，不満点があるとすれば音楽かな？

「シムアース」が発売されたと同時に7位にランクイン。SX-WINDOW対応が高く評価されています。スピードには多少不満の声もあるようですが，オリジナリティという点では抜きんでていた「シムシティー」の思想を受け継いでいるだけに，ゲームとしての出来はいいようです。

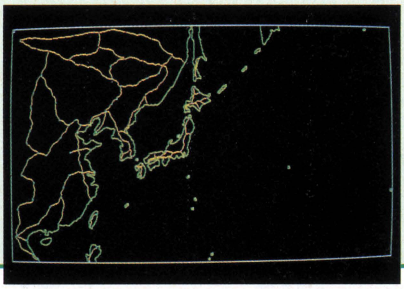
9位にはズームの期待作，「OVERTAKE」がランクイン。F1ゲームというジャンルそのものが待たれていたうえに，作るのがあのズームとあって注目が集まっています。10位タイでランクインした「ポピュラスII」には多彩な天変地異に期待する人が多いようですね。

ではまた来月。トップはどっちだ？　（浦）

バトル

「太平洋の嵐」を発売していたジー・エー・エムの，現代戦シミュレーションゲーム「バトル」がX68000に移植された。“極東の軍事バランスを完全にシミュレート”と銘打つだけあって，登場する艦船，陸戦兵器，航空機の豊富さ，多彩さはスゴイものになっている。また，戦闘だけではなく，各国のイデオロギーや装備の新旧，亡命政権の樹立などもシミュレートされる。

X68000用　3.5/5″2HD版　12,800円(税別)
ジー・エー・エム　☎03(3736)6879

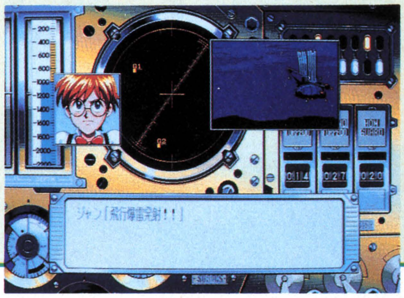


PC-9801版の画面です

ふしぎの海のナディア

PC-9801ではすでに発売されていた「ふしぎの海のナディア」が9月11日にX68000にも登場する。このソフトは，NHKで2年前に放映されていたテレビアニメーションをもとにアドベンチャーゲーム化したもの。謎の海難事故が次々に起こり，戦争の勃発，あいは謎の勢力の登場が噂される19世紀末を舞台に，潜水艦ノーチラス号に乗り込んだ少女ナディアが活躍する。

X68000用　3.5/5″2HD版　14,800円(税別)
ガイナックス　☎0422(22)1980



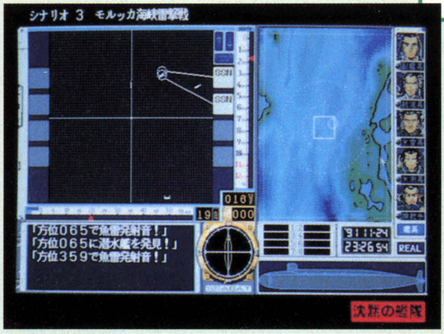
沈黙の艦隊

ジー・エー・エムからはもうひとつシミュレーションゲームの発売が予定されている。あの「沈黙の艦隊」である。原作はかわぐちかいじ氏が週刊モーニングに連載中の同名マンガ。原子力潜水艦「やまと」を巡って繰り広げられる緊迫した人間ドラマが人気を博している。

原作のストーリーは，日本初の原子力潜水艦シーバットが逃亡し，全世界に向けて本艦は独立国“やまと”である，と宣言するところから始まり，各国の政府，首脳を巻き込みながら話は進んでいく。

ゲームは原作で展開された戦闘を，いくつかに分けてシナリオ化している。プレイヤーが演じるのはもちろん，艦長の海江田四郎だ。

X68000用　3.5/5″2HD版　12,800円(税別)
ジー・エー・エム　☎03(3736)6879

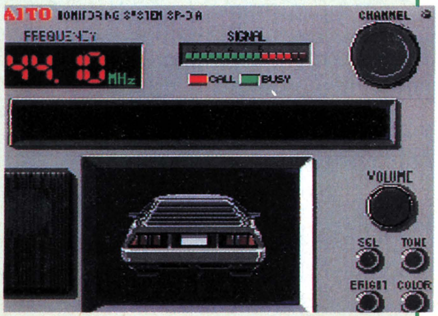


FM TOWNSの画面です

チェイスH.Q

以前，別ロで移植予定に挙がっていた，タイトーのアーケードゲーム「チェイスH.Q」がTAKERUで発売されることになった。このゲームはカーチェイスを楽しむドライビングゲームで，本部からの指示に従い，犯人の車を強制停車させ，逮捕する。移植を担当したのは，これまでにMSXの「ソーサリアン」などを手がけているティールハイトというところ。7月20日発売予定。

X68000用　3.5/5″2HD版2枚組　7,800円(税込)
ブラザー工業(TAKERU)　☎052(824)2493



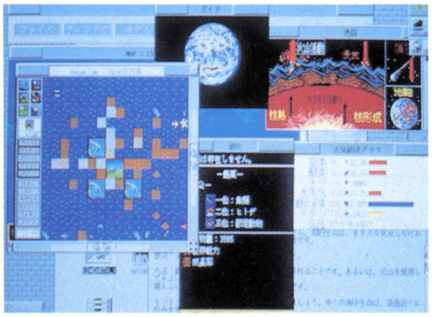
こぼれ話

冒頭でお知らせしたとおり，「ファイナルファイト」は発売中。要2Mバイト，MIDI対応，スペシャルCDインパックで価格は9,800円(税別)となっている。また，「MIRAGE System Model Stuff」も7月4日に発売された。「Communication SX-68K」はそろそろ発売されそう。わりとオーソドックスな仕上がりになっているが，対話形式の自動運転プログラム作成などが特徴。標準ワープロなどに対応した，「FIXER68K ver.4.0 FPコール対応版」も近々出荷される。

TREND ANALYSIS

1992年5月の月間売り上げベスト10

POINT	タイトル	発売元	発売日
499	シムアース	イマジニア	'92/5/22
333	スターウォーズ	ビクター音楽産業	'91/12/17
256	レミングス	イマジニア	'92/4/17
201	太閤立志伝	光栄	'92/5/10
192	グラディウスII	コナミ	'92/2/7
140	棋太平68K	SPS	'92/5/1
138	エイリアンシンドローム	電波新聞社	'92/3/25
130	SX-WINDOWver.2.0	シャープ	'92/3/24
128	苦胃頭捕物帳	電波新聞社	'92/3/25
93	出たな!! ツインビー	コナミ	'91/12/6



今回は諸般の事情により，集計対象店が少なめになってしまった。信頼度は若干落ちることになるが，どうかごかんべんいただきたい。

さて，今月も1位から10位まで，ランクインしたゲームを紹介していこう。

1位は，苦難の末にとうとう発売された「シムアース」。SX-WINDOW対応で要2Mバイト。いや，2Mバイトでも動くといったほうがいいだろうか。心配されたスピードもそんなに遅くないし，当然，ハードディスクにも簡単に入る。マニュアルプロテクトは他機種と同様。

2位には「スターウォーズ」。ランクアップとはいえ，売れ行きが伸びているわけではない。ほかのソフトに好調なものが少ないということだろう。しかし，発売開始から半年たったにもかかわらず，根強い人気を保っているのは確か。

3位は，1位の「シムアース」と同じイマジニアの「レミングス」。見かけはパズル性はそこそこで，アクション的要素が重視されているように見えるが，面が進むにつれ，アクションに負けず劣らず戦略が必要になってくる。華麗なマウスさばきは当然できるものとしたうえで，パズルが考えられている。

4位の「太閤立志伝」は光栄の作品。このところ発売された光栄のシミュレーションゲームは，出るとすぐにそこそこの順位に入ってくるが，次の月には姿を消してしまう。X68000では，固定ファンが発売と同時に買うというケースが大部分を占めていると思われる。しかし，次回は人気シリーズ最新作の「三國志III」が入ってくる。これは動きが注目される。

大人気を博した「グラディウスII」の今月の順位は5位。で，6位が「棋太平68K」。ほぼ同時に「将棋聖天」が発売されているが，残念ながらこちらはランクインしていない。内容的にどちらが優れているか，というのはわかりにくいだろうから，やはり価格で差がついてしまったようだ。ちなみに「棋太平68K」は9,700円，「将棋聖天」は14,800円（ともに税別）である。

電波新聞社の「エイリアンシンドローム」は7位，同じく「苦胃頭捕物帳」が9位になっている。この間の8位に入っているのが，「SX-WINDOW ver.2.0」。九十九電機ではあいかわらず異様に売れている。で，“そこで「シムアース」も多めに入荷してみたら，これもまたバカみたいに売れました。SX-WINDOW対応でなければ売れなかったと思います”とのこと。10位には「出たな!! ツインビー」が復活した。

「SX-WINDOW ver.2.0」にかぎらず，店ごとに順位は相当異なる。ある店で爆発的に売れているソフトが，ほかの店では全然人気がないというのもめずらしくない。それぞれの店の雰囲気，お客の層，入荷状況で，売り上げはかなり変わってくるものだということを毎月の集計で感じる。

［データ集計協力店］（順不同）
九十九電機本店
OAシステムプラザ横浜店
J&P（町田）
P&A

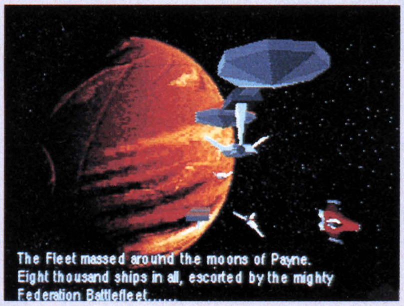
ウワサのソフトウェア（海外編）

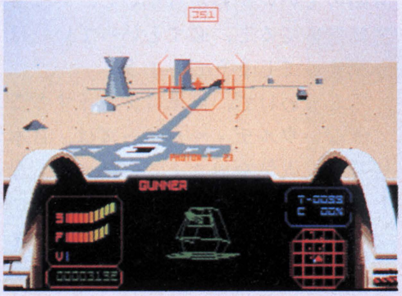
EPIC

ポリゴンによる3D表示，しかも速いとくれば，思い出すのはフライトシミュレータの「F29 RETALIATOR」。その制作元が「EPIC」という3Dポリゴンのスペースシューティンゲームを作っている，というのはずいぶん以前から発表されていて，海外雑誌に広告も出ていた。

まあ，あそこが作るんだから，速いことは間違いないだろうと思って，かなり期待はしていたのだが，ポリゴンの宇宙船が飛び回るオープニングデモが入った販促用のディスクなども手に入り，「うーん，すごいような気がする」と不可解な期待は高まるばかり。

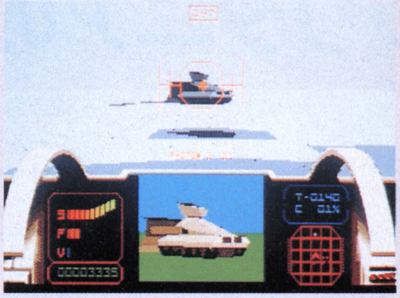
で，やっと発売。で，プレイ。そこは漆黒に星が散りばめられた宇宙空間。自分のまわりには衛星のような，ゴミのようなものが散乱している。もちろん，いつもどおり説明書なんか読





んでいないから，何をどうすればいいのかわからない。「とりあえず攻撃してやれい」とばかりに，まわりのゴミをけちらしていく。こうして宇宙ゴミ処理会社のようなことをしていると，敵の戦闘機がどこからともなく現れ，攻撃してきた。「こいつをやっつければいいのかな」とばかりに宇宙魚雷をぶっぱなす。……ここで撃墜されたり，撃墜したりということを何回かやっていたが，いっこうに先に進む気配がない。時間切れで終わってしまう。パッケージにはポリゴンいっぱいの地上面などが載っているのに，この面は宇宙空間だから何もなくて，見ていてもつまらない。遠くに飾りのような惑星が見えてはいるが，殺風景なのに変わりはない。

なぜ，先に進まないのか。どうやら，敵を倒し，ゴミを破壊したあとで，背景に描いてある惑星に向かって進まねばならないらしい，ということがわかったのは数日後。



で，そのとおりにやってみると，めでたく面クリア。惑星へと突入した自機は，地上面へと進み，やっとポリゴン使いまくりの建造物，敵戦闘機の編隊などに遭遇し，少しは面白くなってきた。ゲーム画面や操作性は「F29」にそっくりだが，地上にぶつかって大破とか，離陸と同時に撃墜されるということはなく，お手軽に遊べるようになっている。

しかし！　冒頭シーンのつまらなさ，とっつきにくさはいかんともしがたい。よって，ガナルカナル島へ遠島を申しつける，というゲームなのであった。オープニングや面と面との間に上映されるアニメーションも同じようなものばかりで飽きてくるし，なにしろそれを見ただけでストーリーがわかるというシロモノではない。ポリゴンもいいけど，演出も勉強してね。

発売元　OCEAN
価格　£29.99

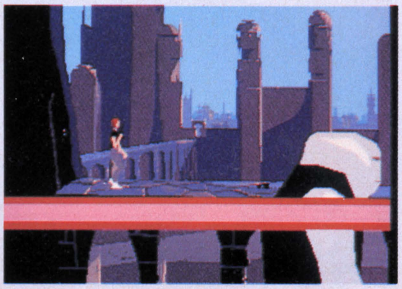
ウワサのソフトウェア（海外編）

OUT OF THIS WORLD

掲載のタイミングを逸してしまって，遅ればせながらの紹介となったが，ひさびさの衝撃。これほどキャラクタの動きで見せることに徹したゲームは「プリンス・オブ・ペルシャ」以来だ。演出の巧みさは映画的。映画の中のキャラクタを操っている気分。ゲームはインタラクティブメディアである，などと柄にもなくほざきたくなる，そんな作品だ。

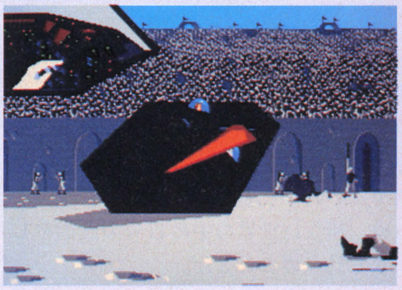
主人公の物理学者が粒子加速の実験中に起こったアクシデントで，別世界に吹き飛ばされるところから物語は始まる。主人公は荒涼とした見知らぬ惑星へ。隙あらば主人公の命を奪おうと徘徊している毒虫。主人公を食べようとつけ狙う飢えた野獣。一瞬でも気を抜けば命を落とす。ここが物語の舞台である。

オープニングからプレイヤーをしっかりとらえて離さない展開。閃光・轟音とともにこの世から消滅した主人公が別の世界に出現した瞬間，もうゲームは始まっているのである。この流れるような導入はあまりにも自然で，プレイヤーを否応なくゲーム世界に引きずり込む。



脱出し，自由を求めて前進。立ちふさがる敵を倒し，仕掛けられたトラップをかわす。アメリカ映画ばりの銃撃・格闘アクションシーンもある。キャラクタは生きているかのごとし。

このゲームが革新的なものになったのは，システムの優秀さのためである。ふだんのゲーム画面はサイドビューで登場人物もそこそこの大きさで表示されるが，ときおり挿入されるビジュアルシーンの構図は映画的で，人物がアップになる。しかし，２次元ポリゴンでのアニメーションなので，絵の質感は変わらない。ビジュアルシーンと通常のゲーム画面とのつながり方が実に自然。そこにギャップは存在しない。



テキストによるストーリー説明は一切ない。それでも演出が雄弁に状況を語ってくれる。

欠点は？　細かいバグはいくつかあるがそれはご愛敬。また，アドベンチャーゲーム特有のきついハマリや高度なテクニックを要求する場面がいくつかあるが，登場人物になりきっているのでそれも楽しむことができる。

しいて挙げれば結末かな。スッキリしたラストではなかった。作者はフランス人だそうだが，アメリカ人なら強引にご都合主義を使ってでもきっちりハッピーエンドにしたんじゃないかな。ただ，この結末はなんとなく続編を予感させるので，いまから楽しみでしかたない。

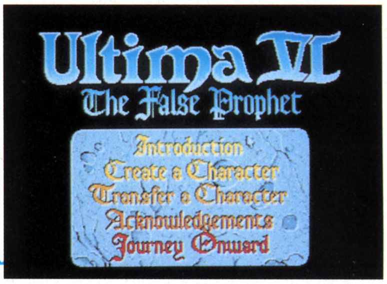
リアルタイムアドベンチャーゲームはまた新たな地平を開いた。万人向きとはいえないかもしれないが，アドベンチャーゲーム史に残るべき偉大な成果である。　(A.T.)

発売元　Interplay
価格　$59.95

ブリタニアの歌声

Taki Yasushi
瀧　康史

海の向こうではとうとう“VII”が出た。さらに，「アンダーワールド」などの亜流も生まれて，ますます盛り上がりを見せている「ウルティマ」シリーズ。X68000にもようやく“VI”が登場し，新たな歴史を築き上げる。



月を見ると，あの頃の思い出が彷彿としてきて，無性に懐かしくなる。月明かりでふと目覚めたとき，ひとつしかない月を見て驚愕することがある。

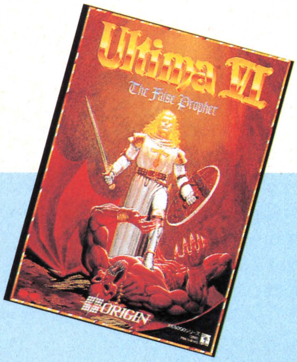
あれから5つの季節が過ぎていた。あれほど望んでいた穏やかな生活と平和，静寂がふと恨めしく思えてしまう。いや，こんな考えは聖者としては失格かな。

ワイングラスに赤ワインを注いで，AVシステムのスイッチを入れた。見飽きたビジュアルばかりがそこにはある。孤独感に悩まされ，つい私が話しかけてしまっても，彼らは答えてはくれない……。

ブリタニアの親友たちと回し飲みしたワインの味。やはりワインはイオロの歌を聞きながら飲むものだ。耳を澄ますとイオロの歌声が聞こえてくる。ブリタニアの思い出が目の前を横切る……。シャミノの話し声が聞こえる。もう一度あの冒険に身を投じたいと思う。傍らに倒れていく友。鮮血！　悲痛の声！　そして，友の死……。

夢？　また同じ夢を見てしまった自分を叱咤する。勇敢とは怪物どもを恐れずに撃ち向かっていくことだけではない。現実を見つめ，新しい世界の中の自分の役割をしっかりと見つめることだ。

そのとき，窓を叩く音が私の思考の邪魔をした。雨か。窓を閉めなくては。今日はあんなに晴れていたのに。



X68000用　5"2HD版3枚組　9,800円(税別)
ポニーキャニオン　☎03(3221)3161

外は雷雨だった。稲妻がなめるように地面を這う。いや違う！　稲妻はストーンサークルの中心を狙って落ちている。もしや，ブリタニアからの呼びかけでは？

TVのスイッチも切らずに，私は家を飛び出した。さっきの回想が走馬灯のように脳裏に浮かぶ。とりつかれたように森を抜け，丘を駆け抜け，体中がびしょ濡れになってもかまわずに走った。

ストーンサークルにたどりついたときには，雨は止んでいた。土の焦げた匂いが漂っている。きっと落雷のせいだろう。

そこにはムーンゲートは開いてなかった。落胆と同時にあわてた自分をあざけ笑う。夢ばかり見ていたから，このような妄想をしてしまうのだ。ブリタニアにはロード・ブリティッシュが帰還してるのだから，私を必要とする要因など何もない。

そういえば家のドアを閉め忘れていた。何もかも忘れて飛び出すなんてバカなことをしたものだ。私はそんなことを考えながら，もう一度見ておこうと振り向いたとき，奇妙なものが目に止まった。

小さな黒曜石らしい。

なぜこんなものが……。そう思いつつ私は手にその石を取る。同時に音もなく光のドアが現れた。鮮やかな記憶が私の体を突き抜けた。あの栄光の輝かしい記憶が。思わず再び石を見つめる。手に汗がにじむ。

しかし，興奮はある疑問とともに立ち去った。ブリタニアではゲートはいつも青かった。しかしこれは赤い……不吉な予感が脳裏をかすめる……。なぜ？

紅色の光の衰退が思考を中断させた。考えているひまはない！　飛び込まねば！

幸福感が私を支配するのを感じる……。

アバタールになった日本人

かくして，アバタール（聖者）である主人公は旅に出ます。

ウルティマを簡単に説明するとシナリオ重視型の大型RPGです。最近の日本の主流となっている盲導犬型のRPGでは，ひとつのクエストが終わると，もうひとつの謎かけが……というように次々とお話が進んでいきますが，ウルティマではそうはいきません。歩けば歩くほどいろんなクエストに出会い，それらが同時に進行します。何から始めていいかわからないし，メモも取らずに進めてると，必ずといっていいほど行き詰まってしまうでしょう。

そんなわけで，簡単にプレイすることができないウルティマシリーズは，日本では爆発的な人気を得るまでにはいたらなかったようです。たしかに，民族性の違いなどを痛切に感じてしまうところもありますが，要は楽しみ方を認識できていないといえると思います。

まずはブリタニア概論（添付される本）をよく読み，さらにゲームのオープニング，イントロダクションなどをじっくり読んで，自分自身がアバタールになりきることから

質問には正直に答えよう

ロード・ブリティッシュは今回も健在

始めます。
キャラクタメイキングは，ウルティマIV以降から採用された独特の徳についての質問で決定されます。どの答えを選んでも間違いではなく，８つの徳のうちの２つを天秤にかけ，どちらの徳をあなたは尊重するか？　みたいな問いでまとめられています。ただ，ウルティマIVのように直接それが影響して，キャラクタの外観（職業）が変わってしまうことはありません。パラメータも変わってはいないようです。おそらく，内部パラメータの徳についてのカルマが変わっているのでしょう。

イントロダクションを眺めていると，なぜかムーンゲートを通ったあと，いきなりガーゴイルに捕まってしまい，生贄として処刑されてしまいそうになります。そこを，かつての仲間である，イオロ，シャミノ，デュプレの３人に救われるわけです。このとき，赤いムーンゲートを通っていくのですが，同時にガーゴイルが３体紛れ込んできます。

ですから，ゲームを始めると，いきなりロード・ブリティッシュの目前で，３体のガーゴイルとの戦闘になります。

とりあえず，このガーゴイルはそれほど強くありませんから，適当に倒してしまいましょう。城にはロード・ブリティッシュ以外にも人がいますが，彼らには攻撃してはいけません。なぜなら，自分自身はアバタールだから，無意味な殺生などをしてはならないからです。

あとは，出会ったキャラクタにいろいろ話しかけてみることで，ゲームは進行していきます。膨大な人数のキャラクタがいますので，当然膨大な情報量になります。多少でも重要だなと思ったら，すぐノートなりにメモしたほうがいいでしょう。

SPEEDとHDとMIDI

さて，今回のウルティマのゲームシステムでは，従来と違って縮小スケール（前作での移動中の縮小マップ）はなくなり，戦闘中も街中も，一貫したスケールとなっています。そのため，戦闘もいきなり，

「敵が現れた！」

となるのではなく，画面上に自分と同じように存在し，徘徊し，徐々に近づいてくるのがわかるというシステムです。

気になる速さですが，10MHzマシンでは多少の重さを感じてしまいます（ふだん速いマシンでやっているせいですが）。ただ，重くてどうしようもないわけではないので，それほど気にする必要はないでしょう。

気になるところは，マニュアル（解説書）をよく読まなくてはゲームがまともにできないところです。シナリオ的にブリタニア概論を見るのは許せるのですが，操作方法がわからずにマニュアルを何度も見るのは，ゲームのノリが崩れてしまって，やっぱり興ざめです。

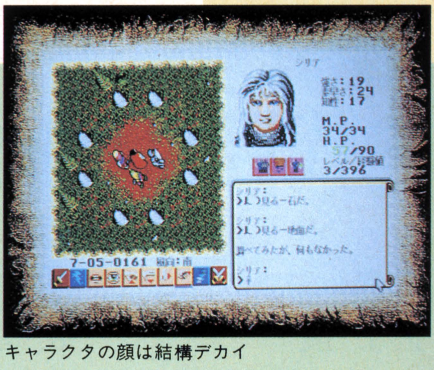
それからソフトウェアプロテクトがなくなり，ハードディスクにインストールできるようになりました。ただし，マニュアルと一緒についてくるブリタニア概論の内容を参照する必要があります。賞賛すべきは，マニュアルプロテクトである質問がシナリオ中に無理なくはまっていて，プレイ感を損なうことなく進めることです。

ただ，不満がないわけではありません。たとえばハードディスクへのインストールなどは，多少のOSの知識が必要になります。また，起動メニューなどから指定するのではなく，いったんHuman68kのディスクを起動したうえで，インストールプログラムを起動しなければなりません。デバイスドライバに余計なものが入っていると，インストールプログラムが途中で暴走してしまうこともあるようです。

新しく対応されたものはハードディスクだけではありません。MT-32系のMIDI音源にも対応しています。もともと，ウルティマの曲はよい雰囲気を醸し出しているので，かなりの臨場感があります。

ただ，MIDIの設定を変えるのに，いちいちOSを立ち上げて，MENUというバッチファイルを起動しなくてはならないところには疑問を感じます。

ムーンオーブでムーンゲートが開く


キャラクタの顔は結構デカイ

クセがあるから面白い

謎解き探求が好きな人にはお勧めなゲームです。民族性の違いなどで，なかなかとっつきにくいところはありますが，慣れてしまえば問題はありません。独特の世界観が底辺に流れているという点では，このシリーズの右に出るソフトはありません。

シナリオはかなり練り込まれていますし，これを機会にウルティマの前作をやってみたいと思う方も出てくるでしょう。

最近，日本では主流となっている，何も考えなくてもポンポン進むRPGに飽きた人には，ぜひトライしてほしいと思います。ゲームを解いたときの喜びは最近の国産ゲームではなかなか味わえないものがありますからね。

移動モードも戦闘モードも画面は変わらない

慣れというのは恐ろしい

ひさしぶりにウルティマをやったら，いつのまにか国産のアクションRPGに慣れてしまったせいか，とっつきづらいものがありました。しかし，いざ適応してしまうと，この探求という作業が楽しくなってしまうのは不思議です。

音楽に関しては，内蔵音源では無難な出来ですが，MIDI版ではかなりいいセンだと思います。標準音色から変えていないのが逆に幸いして，コンパチ音源のSC-55などでも素直に聞くことができるのがうれしいですね。新発売のヤマハのTG100のC/Mモードでもかなりしっかりと聞くことができました。

総合評価	0　5　10
操作性	★★★★★★★
システム	★★★★★★
シナリオ	★★★★★★★★★
音楽	★★★★★★★★★
スピード	★★★★★★
お勧め度	★★★★★★★★★

3代目はクォータービュー

Kaneko Shunichi
金子 俊一

「三国一の婿」は，天竺，唐，日本で一番，つまり世界一の婿ってこと。三國志は，紀元2世紀の中国にあった，魏，呉，蜀の興亡を描いた物語。「三国一の武将」に達するのは，「三国一の婿」になるのと同じくらい険しい道だ。

「三國志II」が本誌を飾ったのが，1990年の6月号。しかも，X1turbo用ときている。実に2年以上ものブランクを経て，いまここに「三國志」が復活したのだ。

中国三千年の歴史

今回は型から入ってみよう。三國志とは中国の正史である。日本でいうならば，日本書紀とか，古事記にあたる書物といえる。時代的には弥生時代で，卑弥呼がいた頃とほぼ同年代から話が始まっている。日本の古墳文化よりも前に，大陸では劉備や曹操が100万の兵を操っていたわけで，なんともはや中国三千年の歴史である。

一般的に日本の歴史は小学生の頃から教えられるので，織田，徳川，関ヶ原などというキーワードは馴染み深いと思われる。ところが，中国の歴史はサクッと流して教わるので，魏呉蜀の時代は教わっても，孫権，赤壁というキーワードは教わった記憶がないだろう（忘れてしまっただけかもしれないが）。できれば小説でもマンガ・テレビでもかまわないから，このゲームを始める前に予習をしておこう。256倍はハマれると思う。

都市と戦場

「三國志III」では新たなる概念が導入された。それは「都市」と「戦場」である。過去の光栄の歴史ゲームでは「三國志」シリーズにかぎらず，統治単位は国レベルであり，白地図を色鉛筆で塗り分けたような勢力図が描かれていた。

これでは史実に残る西塞山だの長坂といった，一大決戦があった戦場で戦っている気分にはなれない。要はイマジネーションなのだろうが，「五丈原で曹軍と劉軍が激突」したなら，きっと感動できるだろう。たとえ自分が他国の君主で，当事者でなくたって，である。それが同じ場所でも「漢中で激突」といわれたら，平凡な戦争と思ってしまうだろう。この意味がわからない人は書物の三國志を読んでみよう。きっとわかるはずだ。

そ・こ・で，いままでのような国レベルでの統治は「都市」になった。都市に対して耕作や治水を行うのだ。

都市と都市は道でつながっており，つながってない都市には攻め込んだり，移動することはできない。国は隣り合っていても，険しい山に阻まれていれば行き来できないのが道理。

そして都市と都市の途中に「戦場」がある場合がある。この戦場を通過できるのは戦場を領有している君主の軍のみである。この戦場を領有するために戦争があるのだ。これは野戦になり，都市を攻めるのは攻城戦である。

都市の数は46あり，戦場は22ある。中国を統一するには都市を制覇すればいいのだが，戦場の制覇なしに統一はありえないだろう。

戦場も赤壁，五丈原，函谷関，官渡，合肥などの聞き覚えのある地名ばかりである。三國志マニアは燃えるぞ，こりゃ。

新君主・春麗

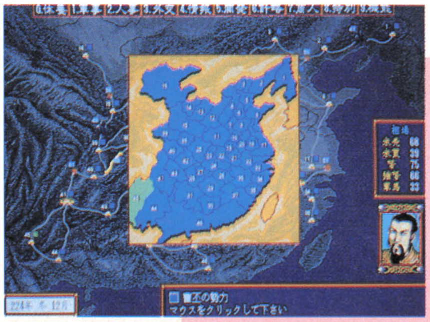
では，マニアでないと燃えないかというと，そうでもない。それは新君主という設定である。マニアならばきっと君主の誰かに思い入れがあるだろう。関羽を仕官させて曹操の思いを遂げさせてあげたかった，などである。

しかし，新君主では自分だけの君主という，別の意味での思い入れができるのがポイントである。これは「三國志II」でもあったが，配下の武将が選べなかったので，スタートが厳しかった。その点，IIIになってからは，スタート時点で配下の武将も3人まで選べるようになったし，登録できる人数も君主が8人，武将が60人となっている。

ここは一発，春麗ちゃんを君主にしてしまおう。配下には軍師・春六（ハルロク）とか将軍・斬消斧（ザンキエフ）などと，ゾッキー（暴走族）も真っ青なセンスで武将を作っていく。

武将は，将軍と軍師以外にも武官と文官という職が登場した。パラメータによって，将軍や軍師になれる人とそうでない人があらわれたのである。だいたい，いままでが数百人単位でキャラが登場しながら，全員

X68000用 3.5/5"2HD版3枚組14,800円（税別）
光栄 ☎045(561)6861



中国大陸の形は当然ながら変化なし

生かすも殺すも胸三寸

が将軍ということ自体に無理があったのだ。五虎将軍と足軽大将が同じ位にいるようなもの。今回は登場人物が約500人にのぼり，さらに新武将も登録できるのだから，能力によって位が変わっても当然といえる。

武官や文官では，それぞれが使用不可能なコマンドというものが出てくる。たとえば武官なら治水などの開発系ができない，文官は戦争に行けない，などである。もちろん，将軍や軍師はどちらもできる。

ほかにも違いはいくつかある。たとえば，文・武官を都市の大守に任命すると，その都市は無条件で委任された状態になる。逆にいえば，将軍・軍師でないと直轄はできないのである。

ありがたい孫子の兵法書を発見

どの武将にするかな

HEX戦線変化あり

戦闘シーンは従来のHEXのラインが消えて，ちょっとミーハーなクォータービューになっている。実際にプレイした感想では，特に秀でているわけではないが，見た目がかっこいいので許してあげよう。

攻撃方法では一騎討ちというのがあるが，これは実によい。2人プレイにすれば，呂布と関羽ではどちらが強いか，などとほとんど紙相撲のようなノリで遊べてしまう。自分が作った春麗は張飛には勝ったが，七星剣を持った趙雲には負けてしまった。くやしい。

一般兵士は歩兵，騎馬隊，弩，強弩と4タイプに分かれ，それぞれ移動力や攻撃射程などが違う。「信長の野望」シリーズではすでに採用されている手法であるが，好評だったのだろうか，「三國志III」でも採用された。

また，水域では船による戦いが再現されている。魏VS呉蜀連合軍という中国大陸を二分した赤壁の戦いでも，呉の周瑜が率いる水軍が大活躍をしていた。「三國志」に船は必需品である。

船は3タイプに分かれ，闘艦，蒙衝，走舸となる。現代風にいえば，それぞれ戦艦，駆逐艦，水雷艇のようなものだ。機動力が違うのだが，闘艦のほうが低いわりに値段は高い。おそらく，船に乗れる人数が違うと思うのだが，マニュアルには書いていない。これなら安くて速い走舸がイチ押しといってしまうぞ。

ちなみに，馬，弩，強弩は商人から買うことができるが，闘艦，蒙衝，走舸は自分で建造しなければならない。建造にはそれぞれ6,4,2カ月かかる。この手法は古くは「ディーバ」（T&E SOFT）がやっていたと記憶している。

とってもコンビニエンス

さすがに3代目ともなると，細かい部分での気遣いが増えている。ユーザーの声の反映というやつか。たとえば，武将の選択ひとつでも大違いなのである。

治水をするときと民に施しをするときでは，コンピュータが参照するパラメータは違ってくる。治水では「政治」が重要で，施しでは「魅力」がポイントになる。それぞれの値が高いほうが効果は大きい。

しかし，ゲームも後半になってくると，すべての武将のパラメータなんてとても覚えていられない。ターンごとに武将のパラメータを見るなんてナンセンスだし。そこで，このゲームでは武将の選択のときに，そのコマンドに関連するパラメータを表示してくれるのだ。プレイヤーはそれを参考に，最も適した人材に任せればよい。考えただけでも便利そうだが，実際には予想以上に便利。

ほかにも便利なところがある。ロード/セーブには専用ディスクがいるので、本来ならディスクの抜き差しが必要だが，立ち上げ可能なユーザーディスクにもひとつだけセーブできる。これでディスク交換なしでロード/セーブができるようになった。

足らない能力をアイテムで補うことができるようになった。孟徳新書では知力と政治力が，青竜偃月刀では武力が，赤兎馬では機動力がそれぞれプラス修正される。もちろん，あり余る能力をさらに伸ばすことにも使える。ほかにもアイテムはあり，孫子の兵法の書とか七星剣といったものや，けがの回復といった効果のものもある。

締めのお言葉

素直に面白い。特に三國志マニアはぜひチャレンジしていただきたい。パンピー(いっぱんぴーぷる）でも面白いと思うので，シミュレーションを考えている人にはハナマルでお勧め。

X68000らしさという点では，画面のドット数の関係から，戦場でのパラメータが見やすくなっている（PC-9801比）。また，めずらしくディスクをイジェクトしてくれるときもあるようだ（ほとんどはしてくれないが）。これからは，ちゃんとイジェクトの認識してよね。

戦闘場面はかなりかっこよくなった

辛口批評

今回は3作目ということで，あえてシミュレーション周りの解説を入れなかった。記述されていない点は，ほとんど従来どおりのシステムと同じと思っていただいてかまわないだろう。

IIに続いて音楽はカシオペアの向谷実氏が担当している。作曲を担当したのか，コーディングまでやったのか，音色はどうしたのか，など聞いてみたいことは山ほどあるが，内蔵音源で聞くかぎりでは特筆するようなレベルではない。おそらく，楽譜からのベタ起こしなのではないだろうか。ちなみにMIDIには対応していない。

また，ディスクアクセスが多いのはマイナス要素。武将の顔の表示や季節の変わり目などでアクセスする。メモリが許すかぎり読み込むのも手だが，ここは正統にハードディスクへインストールできるようにしていただきたい。これほどマニュアルプロテクトに向いているゲームもほかにないと思うのだが。

総合評価（0〜10）

ゲームシステム	★★★★★★★★★
グラフィック	★★★★★★★
ミュージック	★★★★★★
演出	★★★★★★★★
ユーザーインタフェイス	★★★★★★
一騎討ち	★★★★★★★★★

メックとメックでバトルテック

敵を倒しながらお金を増やし，装備を強力にしていく。自分の経験値を増やすとともに，頼れる仲間も見つけていく。これだけなら「もういいです」といいたくなるかもしれないけれど，戦闘がポリゴンで表現された3Dなら……。

Kageyama Hiroaki
影山 裕昭

もともとバトルテックというのは海外のボードゲームで大ヒットしたシリーズだそうだが，恥ずかしながら私はそのブームにまったく気づかなかった。ひと昔前のガンダム，現在のドラクエのような人気なんだろう。日本でもバトルテックを題材にした小説や，ロボットのコクピットをシミュレートしたゲーム機を8台リンクして対戦できるようにした“バトルテック・センター”が今春オープンしたそうだ。そして，X68000には『バトルテック～失われた聖杯～』が発売された。

綿密な舞台設定

最近は舞台背景がしっかりしているゲームが多いが（肝心なゲーム内容でコケているソフトも多い），このバトルテックも例に漏れず舞台背景がやたらとしっかりしている。ボードゲーム出身の面目躍如である。舞台背景を説明するだけでレビューが終わってしまうほどだ。かいつまんでいえば，

「時は西暦3024年。主人公であるプレイヤーは名家バンデンバーグ家に育ち，世襲でアンダースムーンと呼ばれる星の君主になる予定だった。が，君主の象徴であるハーンの聖杯を対立するマクブリン一族に奪われたうえ，マクブリン一族の陰謀でアンダースムーンを追い出されてしまった」

という設定である。略奪者たちについて確認できた情報は，翼に縁どられたドクロの紋章を付けたメック（モビルスーツ）を操縦して逃げた，ということだけである。奪われた聖杯は5つある星系の中のどこかの星に隠されている。プレイヤーは傭兵となり1機のメックといくらかの金を持って，星から星へ旅をして情報を集め，仲間を集め，マクブリン一族から5年以内に奪われた聖杯を取り返すことがゲームの目的である。

傭兵としての心得

5つの星系は，31世紀に人類が宇宙空間に築き上げた，スターリーグと呼ばれる一大星間帝国が分裂した結果誕生した。ひとつの星系に20～30個の星が属し，数えてみると全部で145個も星がある。星間移動は金がかかるが，どこにでも自由に行くことができる。

ひとつの星でとれる行動は大きく分けて，個人情報/ニュースネットの閲覧，メックの整備/購入，星間移動，酒場での情報収集/傭兵の雇用，任務契約，オプション（データの保存，読み込みなど）である。個人情報というのは，所有金額や年齢，5つの星系の領主がプレイヤーに対してもっている感情が“最高”から“最悪”までの5段階で表される。感情が最悪の星系では，まず任務の契約をすることができないだろう。ニュースネットでは，最近の主だった事件を閲覧することができる。ここで重大な情報を得ることが多々ある。

情報収集の場としては酒場も重要である。酒場では情報だけでなくメッククルーを4人まで雇うことができる。クルーを増やしておけば，多少厄介な任務も契約が結べるようになるだろう。しかし，雇ったクルーはメックを所持していないので，メックはこちらで買い与えることになる。だから，最低メック1機＋アルファの金額を所持していないと雇っても意味がない。

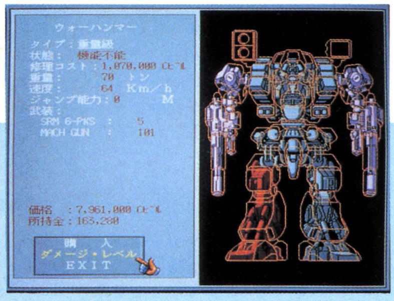
星によっては（特に他の星系との境界線に近い星）傭兵を必要としているところもあるので，そこでは傭兵として任務を契約することができる。契約にあたっては慎重な姿勢で臨むようにしたい。なぜならプレイヤーの主な収入源は，契約した任務の成功報酬だからである。

“楽して金を稼ぎたい”とは人間誰しも思うことだろう。任務にも楽な任務，辛い任務がある。私の経験上，楽な任務は「反乱軍の鎮圧」である。これは敵のメックをすべて破壊した時点で任務完了である。反対に辛い任務は「長期遠征」である。その名の示すとおり長期の戦闘を行うものである。弾薬補充やメック修理のために基地に戻ることができないので，メックを自在に操る操縦技術と効果的な弾薬の使用法を知らないと契約完了は困難だろう。

そのほかの任務には「人質救出」「護衛部隊任務」などがあるが，どれがおいしい仕

アイコン形式のメインメニュー

戦闘シーンは3Dで表示される



X68000用 3.5/5"2HD版2枚組 9,800円(税別)
ビクター音楽産業 ☎03(3423)7901

事なのかは，実際にその任務を請け負ってみるまではわからないだろう。しかし当たり前の話だが，楽な仕事ほど報酬は少ない。だからといって悲嘆するのは早い。契約の交渉ができるからだ。ここは大阪商人になりきって，思い切り金額をつり上げるべきだ。交渉の成否は，その星が属する星系へのそれまでの貢献度で決まる（貢献度は前述した領主の感情で判断できる)。あまり交渉が長引くと契約を中止されてしまう場合もあるから，ぎりぎりの引き際を見きわめるのが肝心である。

傭兵の旅立ち

さて，最初にやっておくべきことは傭兵の仕事道具でもあるメックの整備だろう。与えられたメックが正常であるとはかぎらないし，弾薬が装塡されていないかもしれないからである。行動はアイコン形式のメニューをマウスカーソルでクリックして決定するのだが，このゲームではマウスよりもキーボードで操作したほうがゲームの進行がスムーズである。なにもマウスカーソルの反応が悪いということではないが，感覚的にそういうことなのである。

整備工場では設定資料集のようなメックの詳細図が表示され，故障箇所が赤色で表示される仕組みになっている。画面写真を見てもらえばわかるだろう。これには驚いた。たいして役に立つ画面ではないが，ゲームに対するこだわりがうかがえるし，なによりもイマジネーションの育成にひと役買っている粋な演出である。故障箇所があればメックを完璧に直しておこう。

やることがないので，酒場に行って“翼に縁どられたドクロの紋章”についての情報収集をしてみる。バーテンダーがいくつかの情報を話してくれたが，どれもあまり役に立ちそうではない。ま，それでもあとで役に立つかもしれないからメモを取っておこう。フムフム，グレーデス集団……。まだメッククルーを集めるほど資金がないので，聞いたことをメモしたら酒場をあとにしてかまわないだろう。

メックを完璧な状態にしたら，任務を契約することにしよう。最初は簡単そうな任務を契約したほうがいいだろう。もちろん交渉して報酬をつり上げることも忘れずに。初めての実戦でメックの操縦に自信があるという人はいないだろうから，オプションの練習モードでメックの操縦に慣れておくことを勧める。ちなみに契約を結ぶと，任務が終わるまで途中経過をディスクにセーブできなくなるから注意が必要だ。それから，星間移動する前にもセーブしておいたほうがいいだろう。

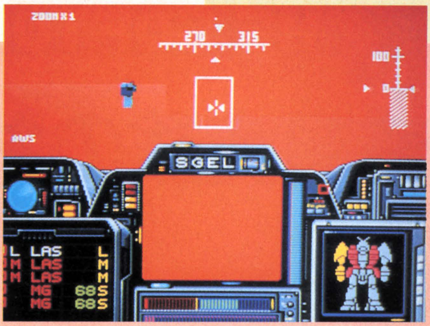
さて任務を契約したら星間移動して任務地に向かう。戦闘前は戦場の簡単な地形図が表示される。メッククルーがいるなら二手に分かれて，１機が正面から，もう１機は山陰から回り込んで背後を攻めるとかいった作戦を立てることもできる。が，そんな面倒なことはせずに，正面から攻めてしまってもいい。コツさえつかめば無傷で敵を倒せるからね。

戦闘シーンは敵味方のメックが入り乱れてのポリゴン表示であるから，その迫力はすごい。表示速度が遅いのが残念である。表示物体数が多くなるとキーの受けつけが悪くなって少々イラつく。オプションで山などの障害物を表示しないようにすることもできるようになっているのは，ユーザーにとってはうれしいかぎりである。

メックの操縦は簡単だ。主に使うのは，テンキー（前後左右の移動)，スペースキー（武器の発射)，TABキー（持っている武器を一斉射撃)。最初は慣れないかもしれないが，２，３回の戦闘をこなせば，難しいことはなくなるだろう。敵メックの胸の装甲は厚いので，コツとしては足を狙うこと。そうすると，相手はこけて動けなくなるので，なかなかおいしい戦法となる。

戦闘を重ね，傭兵としての技術が上がるにつれて，たくさんの情報が入ってくる。情報を手に入れたら，その情報に関係のありそうな星に足を運ぶことだ。そうすると自然と次にやるべきことがわかってくるだろう。それと，こまめにセーブすることも大事だと思う。「名家に育った若者が罠にハメられ地位も財産も失い，一度は絶望の淵に立たされるが，その苦境を乗り越え最後はヒーローとなる」というありがちなシナリオだが，こうして次々と目的を達成し，ゲームを終了させたときの充実感は素晴らしいものだった。

任務の契約は慎重に



攻撃は受けないほうがいいんだけど

目的地を選択して星間移動

情報やクルーが必要なら酒場へ

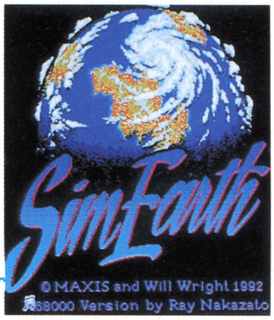
傭兵の皮を被った君主

FLOATがver.1.0なのには少し疑問を感じる。それとディスクアクセスが多い。メモリを増設しているユーザーのためにRAMディスク，もしくはハードディスクにデータを転送できるようにしてあると，より快適にゲームが楽しめるのではないかと思うのだが。

こういったゲームでは，決められたシナリオに振り回されているという印象をプレイヤーに与えてはいけない。バトルテックでもシナリオはあるが，プレイヤーに選択権を与えるシーンが数カ所ある（たとえば，殺し屋に襲われたとき，攻撃するか，逃げるかなど)。このときの行動次第で，展開が変わってくる。そういうこともあって，シナリオに縛りつけられているという印象は少なかった。

総合評価	0　　5　　10
シナリオ	★★★★★★★
ポリゴン表示	★★★★★★★
サウンド	★★★★★★★
操作性	★★★★★★
熱中度	★★★★★★★

惑星実験のお時間です



Ogikubo Kei
荻窪　圭

先月に引き続き，「シムアース」で星を育てる。いろいろとややこしいパラメータがありそうだし，何をすればいいのかがわからないかもしれないけれど，あんまり気にせずに好き勝手にやれば，楽しめるのではないかな？

さあて，シムアースしよう。Ctrl+OPT.1+DEL。DINIT.SYS[1] が立ち上がり，デバイスドライバファイルが画面に並ぶ。そこから，NOFEP.SYS[2] を選ぶ。起動する。FSX，SXWIN，と。無味乾燥なSX-WINDOWが起動する。デスクトップにはSIMEARTHのディレクトリが開いている。SIMEARTH.Xをダブルクリックする。“フホヘホハホヘホ”とノーテンキなBGMで，オープニング。Xアイコンが，とたんに黄色くなる[3]。

ええい，何やってんだ，俺は。これじゃ，「ディスクをセットしてリセット」するゲームとなんら変わらんじゃないか。くそ。

メモリ２Mバイトの悲しさよのお。ああ，２Mバイトユーザーのみなさん，シムアースは２Mバイトでも遊べますが，制限は出てきます。ご愁傷様。しようがないといえば，しようがないのだが。

さて，気を取り直して，SX-WINDOWの1024×1024の広大な実画面。フルに使えばシムアースも楽々。やはり，スクロールオン[4] にかぎるのう。

火星のタイムスリップ

シナリオは８つで，ランダム，アクエリアス，石器時代，５億５千万年前の地球（カンブリア紀），1990年の地球，火星，金星，デイジーワールドだ。それぞれに難易度が４段階。難易度のポイントはエネルギー。シムシティーでいう“税金”だ。金がない代わりに，エネルギーが設定されている。惑星をいじる。火山を爆発させる，地震を起こす，植物を生やす，動物を発生させる，地形を変える。どれも大量のエネルギーを消費する。使えるエネルギーに制限があるほど，むずかしいわけだ。うまく成長すればエネルギーは増えるし，失敗すると減っていくばかりとなる。

まあ，最初は“実験モード”がいい。使えるエネルギーは無限大。果てしないエネルギー，戦えば勝つ，必ずってやつだ。気分は「宇宙怪人ゴースト」だね。あ，知らない？　ハンナ&バーバラの1960年代のアメリカン子供向けアニメ。最近ビデオ化されたから，ちょっと話題にしてみました。

ともかく，最初は実験モードがいい。いや，実験モードだけで十分楽しめる。

ここでどのシナリオを選ぶか。最初に“火星”に惹かれてしまうのが成長期にSFを読んだものの悲しさ。火星のタイムスリップ，火星年代記，火星人ゴーホーム。特に，ディックの長編に繰り返し現れる火星の植民地というコンセプト。光瀬龍の「カナン5100年」とか神林長平の「あなたの魂に安らぎあれ」なんかも火星だったし，とにかく，たくさんあることに間違いはない。

火星のシナリオはこうだ。

「あなたはガイアナイザに指名されました」。ガイアナイザってなんだ，ってのはどうでもいい，とにかく，火星を人の住める星にしてやればいいのだ。

このシナリオは使える技が決まっていて，惑星の地殻モデルや大気モデルをいじったりはできない。残念である。この２つを使わずに，火星を温暖化しなければならないのだ。気分は「追憶売ります」じゃなくって「トータルリコール」だ。

始めると，どーやればいいか，説明が出る。なになに，二酸化炭素を大量にばらまけば温室効果で温暖になるだと。さらに，氷隕石を落とせばそれが溶けて海になるだと。面白えじゃねえか。

ほれほれ，ってなもんで，火星のいたるところに二酸化炭素発生装置をばらまく。ガイアナイザはいろいろな発生装置を使えるのだ。たとえば，植物を発生させるバイオ工場，酸素発生装置，窒素発生装置，水蒸気発生装置などもある。

でもって，二酸化炭素の大気ができ，温室効果でなんとか零度近くまで気温が上がったら，お次は海だ。氷隕石を落とし（ガイアナイザは隕石を落とすのもおちゃのこさいさいなのだ），海を作る。

惑星らしくなってきた。でもって，海にバクテリアを発生させ，地には平和を，じゃなくて，バイオ工場で森林だらけにする。ふふ。これでいいぞ。

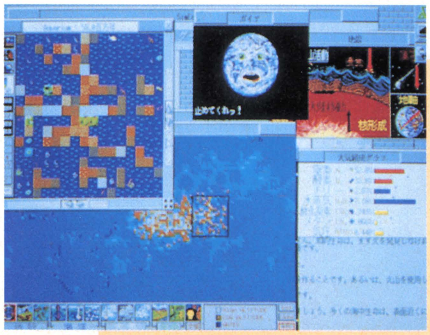
ランダムプラネット

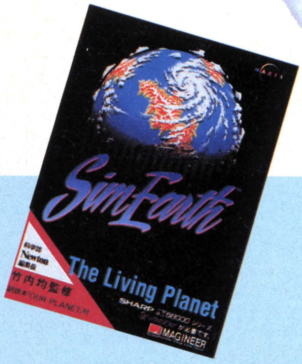
火星の話はこのへんにして，ランダムプラネットをネタに，シムアースの基本をお話ししよう。

まず，メインのウィンドウ。ファイルとかグラフというメニューがあるやつね。

で，マップウィンドウがある。ここには，惑星全土が表示される。惑星のどの状態（地形，温度分布，植物分布などなど）を表示するかは自由だ。

でもって，マップウィンドウの右下にある小さい３つのボタンのうち，GLOBEってやつをクリックすると，惑星全土が球形


いろいろなウィンドウがずらりと並ぶ



X68000用　5″2HD版　14,800円（税別）
イマジニア　☎03（3343）8911

で表現され，回転を始める。惑星儀ってやつだね。これにするとかったるくなるから，普通の人はやらない。

さらに，プレイヤーがいろいろと手出しをするためのエディットウィンドウがある。当然，惑星全土を見ることはできないので，スクロールさせることになる。私はこれのおかげで，新感覚スクロールバーにやっと慣れることができた，というくらい。

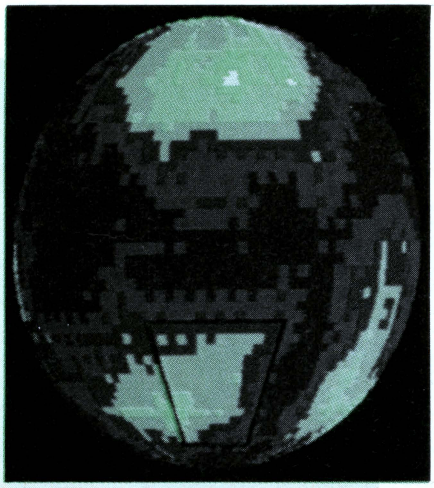
ここでは何でもありである。地表の上下。生命体の発生。各種装置の取り付け。文明の設置。植物の配置。嵐，火山，地震，雷，火事……って具合だ。

さらに重要なのがモデルウィンドウ。地殻モデル，大気モデル，生命モデル，文明モデルを変更できるのだ。こいつはメニューのモデルから選ぶ。地殻モデルなら核熱の調節，地軸の傾き，火山活動などなどを，大気モデルなら熱の反射，温室効果の強さ，雨量などなどを，生命モデルでは繁殖力や進化の強さ，突然変異の頻度などを，文明モデルではその文明の依存するエネルギーや科学・芸術・哲学・農業などの発達度合いを決めることができるのだ。こいつらを自在に操れるようになったら一人前だが，ちょっといじっただけで「大規模な種の絶滅」が起きたりして，むずかしい。逆に，いちばん面白いところともいえる。

あとは，グラフとかヒストリーとかレポートとかあるけど，これらは進化の進み具合やら現在の様子やらをチェックするためのものね。

タイムスケール

これで操作はわかったわけだが，シムアースではタイムスケールという時間の流れに注意しておく必要がある。進化のフェーズが４段階に分けられているのだ。



GLOBE ウィンドウ

まずは，地質タイムスケール。惑星の誕生から多細胞生物の誕生までである。1000万年単位で時は進む。

続いて，進化タイムスケール。知的生物が登場するまでの時代だ。知的生物っていうのは，生物が進化して，知能をもった時点で誕生である。クジラが進化すればエコロジストがよろこぶし，節足動物が進化すれば新幹線に乗り，両生類が進化すれば半魚人の誕生である（いかん，また脱線した)。進化タイムスケールではだいたい50万年単位で時が進む。この時にレポートウィンドウを開くと，どの生物がいちばん進化しているかがIQで表示されて楽しい。

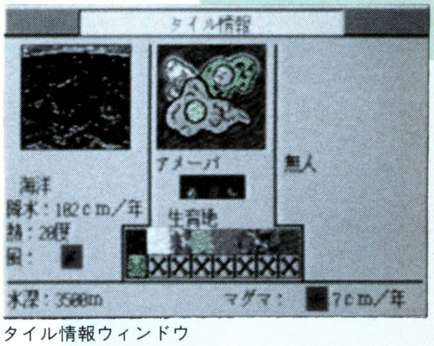
どれかが知能をもつと，いよいよ文明タイムスケールである。これは産業革命までを区切りとする西洋的価値観タイムスケールだ。単位は10年だそうである。

産業革命が起きると，技術タイムスケールである。産業革命は公害を生み，公害は原子力を生み，原子力は核の冬を呼んで，核の冬は情報世代を生む。情報世代は環境破壊を生み，環境破壊はナノテク世代を生んで，ナノテク世代はエクソダスとかいって汚れた地球を脱出する。これで１サイクル終了だ。さらに続けると，再び地質タイムスケールである。で，えんえんと続くかと思いきや，そうではなく，惑星もガイアっていう生物ならば寿命があり，死滅する。ゲームオーバーは惑星の死滅である。

つまり，だ。のほほんとしていても，地質タイムスケールでは光陰矢のごとしで時は進み，技術タイムスケールではゆっくりと時は進むわけだ。これは慣れるまで混乱するが，そもそも地質タイムスケールで１年単位で計算された日にはいつまでたっても生物は進化しないし，技術タイムスケールで何百年って単位で計算された日には，いつのまにか世界大戦でドカンだ。

と，こんなふうに遊ぶ。えっと，ランダムプラネットだな。ランダムプラネットを選ぶと，“この４つのタイムスケールのうち，どれで遊びたい？”ってなことを聞いてくれるわけだ。生命誕生の様子から見るなら地質タイムスケール。ヘンなやつが知能をもったら面白いな，って思ったら進化タイムスケールっていう具合に選べる。

でもって，地質タイムスケールからつらつらと遊ぶ。なんてやってると，なかなか進化しない。およよ，ってコマンドを見ると，“モノリス”ってのがある。モノリスっていうくらいだから，進化させたいやつらに触れればいいんだな，と，触れてやったら，急に知能をもっちまって。“科学技術にエネルギーを注いでください”。注いでやってもあまり変わりはない。しゃあないと，またモノリスの出番。あれよあれよと産業革命，公害続々，原子力革命，チャイナシンドロームばんばん，情報革命。ほーやれほ。いつのまにかほかの生命は死滅し，バクテリアと人類だけが棲息する惑星になっちまった。どーしようかと思って，またモノリスすると，あれま。エクソダスとかいって，地球を脱出しはじめやんの。困ったねえ。破壊するだけ破壊して脱出だって。まあ，いいさね。さようなら～。



タイル情報ウィンドウ

1）　DINIT.SYS：起動時に組み込むデバイスドライバを選択できるフリーウェア。デバイスドライバ名を並べたファイルを８つまで作れる。NIFTY-Serveで見つけた。

2）　NOFEP.SYS：FEPも何も組み込まない，ひたすらフリーエリアを確保したいときに使うファイル。DINIT用に作った。

3）　SX-WINDOWではメモリ残量が足りなくなると，Ｘアイコンが緑から黄色になる。赤になると終わり。

4）　スクロールオン：SX-WINDOW ver.2.0では実画面モードが使える。1024×1024の画面のことだ。で，スクロールオンにすると，マウスポインタが画面の端に行くと自動的にスクロールしてくれる。スクロールオフだと，してくれない。デスクトップ上で右ボタンを押すと，メニューが現れる。

てなわけで

シムアースは２Ｍバイトだって10MHzだって楽しめるのであった。むずかしいことを考えちゃいけない。感動は知識になんてならない。遊びは遊び。それが知識になるかならぬかは，遊びながら感動しながら，それと同時に何をしていたかにかかっているわけだね。

シムアースはちゃんと考えてやろうと思うと，非常にややこしい。でも，ちゃんと考える必要なんてない。複雑な因果応報とともに我々はある，ってことがわかれば，細かい知識はどうでもいい。

私はといえば，カンブリア紀の地球っていうシナリオで遊んでいたら，クジラが知能をもってしまったもんで，それで楽しんでいる最中だ。しかしなあ，クジラの産業革命って何なんだろう。あ，飛行機が飛んでる。あれにクジラとかイルカのなれの果てが乗っているかと思うと，また感動もひとしおなのであった。

AFTER REVIEW

今月は皆さん待望のグラディウスIIです。賛美の声だけでなく，シビアな意見を探してみたりもしたのですが，まったく見当たりませんでした。それだけこのゲームが皆さんの支持を得ているといえるのでしょう。

グラディウスII

▶名作はこれだ！　といわんばかりのグラフィック，ミュージック。特にサンプリング音のクリアなこと。X68000版のために模型まで作ってるし，力の入り具合が違う。
田辺　和也(17)神奈川県

▶ゲーセンの移植だから。そしてグラディウスシリーズのなかでいちばん気に入っているゲームだから。笹木　孝則(17)千葉県

▶レーザーがちょっとあれだけど，正直いってすごい。　阿部　学(19)埼玉県

▶BGMが特にイイ。実は僕，ず〜っと前にこのゲームのアーケード版のBGMが収録されたCD「SPACE ODYSSEY GRADIUS II GOFERの野望」を買っていたんですよ。その頃から何回も聴いていて，カッコイイBGMだなぁ，と思っていたんですよ。それが，X68000で聴けるのだから，超感動です。　三沢　弘之(20)神奈川県

▶アーケードではまっていたから簡単と思っていたらなんと！　安全地帯が変わっていた。ショック！　それにしても2周目はアーケードよりムズいぞ。なんで？
服部　直幸(18)長崎県

▶おもしろい。あれだけ敵が出てくるのに，遅くならないのがよい。
熊下　泰章(16)岩手県

▶グラディウス，沙羅曼蛇，パロディウスだ！ を経て，ついにグラディウスIIがX68000でプレイできるようになりました。僕みたいなグラディウスシリーズをプレイしたいがためにゲームセンターに通っていた者にとっては，めちゃくちゃ感動モノの出来事です。ところで，このグラディウスIIのVERY DIFFICULTモードの2周目以降をX68000 XVIの16MHzモードでノーミスクリアできる人，誰かいます？
斉藤　法男(21)滋賀県

▶しばらくこれを超えるソフトは出ないでしょう。それにしてもカニ道楽は強い。
本田　真介(21)熊本県

▶ゲーム自体の完成度の高さはいうまでもないが，X68000版となっていたれりつくせりの機能がアップされている。
久保　誠(28)京都府

▶グラディウスIIはおもしろい。毎日やってもあきない。お気に入りは3番装備。EASYランクならやっと1周クリアした。しかし1番，2番装備はとっても難しい。レーザーが敵に当たらん。とにかくコナミさん，えらい。次はライフフォースだ。
上田　孝一(21)福岡県

▶いろいろな攻略法があって，何度も楽しめるから。　松永　正弘(21)京都府

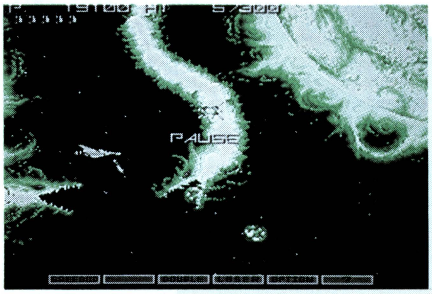
▶やっぱコナミ一番。あの透きとおったような音楽が好き。はやくMIDI一式揃えたいものである。　新井　政樹(20)千葉県

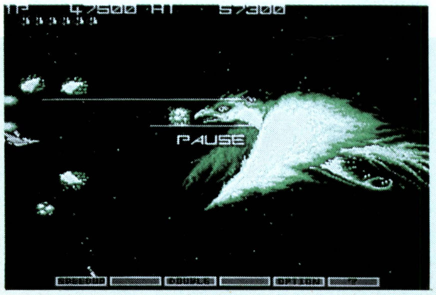
▶グラディウスIIはすごすぎる。もう最高！　でもさすがにまともにクリアできない。USAモードでコンティニュー連打連打……。そのうちゲーセンでノーミスクリアできるぐらい練習するぞ（置いてある店は少ないけど……）。　前田　光(20)千葉県

▶プレイ感覚はアーケード版そのもの！やはりX68000とグラディウスは切っても切れない？　阿部　秀幸(18)北海道

▶友人の買ったグラディウスIIをやらせてもらった。しかし，私には火山の面までしかいけなかった。パロディウスだ！ は最後までいけたが，グラディウスIIは最後までいけそうにない。これでレベルは普通などといわれているのだから，グラディウスIIIはおろか，これから発売されるシューティングゲームは簡単には買えなくなった。やっぱり買うからには最後までいきたいですからね。　山本　昭治(23)神奈川県

▶出たな!! ツインビーやパロディウスだ！もよかったが，それを超えている。単なる移植にとどまっていないのがよい。
横尾　健一(22)三重県





▶内容もすごいけれど，ゲームに添えられた演出（ハードディスクインストール，オンメモリ可，MIDIの選択など）を高く評価したい。　中元　昌文(16)兵庫県

▶ハードディスクインストールでドライブが指定できない以外，文句のつけようがないっ！　信太　徹(21)神奈川県

▶クソゲーつかまされた思い出も吹っ飛ぶデキのよさ！　なんだか来年のGAME OF THE YEARが早くも内定したみたい。
小薮　賢(22)埼玉県

▶ゲームセンター，パソコンをあわせても，もっともハマったゲーム。最近はゲームセンターから消えてしまって悲しいかぎりだったが，まさかX68000に，ここまで完璧に移植されようとは……。カンドー。
伊藤　千光(18)千葉県

▶X68000とは相性がいいからグラディウスIIIも出してほしい。ゲーセンで腕前を見せてやりたかったが，グラディウスIIはもうない……。　対馬　健治(25)青森県

▶とっても出来すぎ。X68000にはおずりしてしまった。　内田　好則(19)長崎県

▶待ちに待ったグラディウスII，発売が発表されたときには狂喜，手にしたときには涙を流しました。ああ，出たな!!　ツインビーに続いて，あのグラディウスIIが自宅でいやというほどプレイできるなんて，X68000ユーザーはなんて幸せなんでしょう。　林　久(20)神奈川県

▶僕はいままでのX68000のゲームのなかで，最高の移植度だと思う。
山田　聖治(18)静岡県

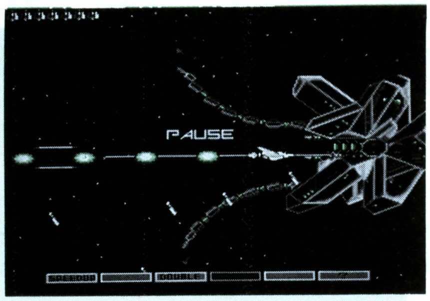
▶X68000では考えられないほどの移植度，すばらしい。ただひとつ難をいえば，アーケードモードでクリアしても，エンディングメッセージがX68000モードと同じなのはいやです。　松本　健一(17)静岡県

▶やはりいま現在はこれを推薦しなければ。多少不満もありますが，よくできています。
中村　健(22)埼玉県

▶こりゃしゅげー!!　思わずパソコンショップでハマってちょっと休んでたら，ジョイカードが抜かれてた。
松本　康裕(24)広島県

▶やっぱりシューティングゲームをやる人にはぜひ買ってもらいたい１本です。アーケード版でいくらつぎ込んだことか……トホホ。　森崎　剛(19)広島県

▶グラディウスIIを買いました。リップルレーザーがアーケード版と比べて弱くなっている，マイナーな安全地帯は再現されていない，最終面の中ボスがモノトーンである（ショック！）など，ささいな不満もあるけれど，一時期は不可能とまでいわれたX68000版を出してくれたコナミに感謝！　ドキュメントによると，ゼクセクスを開発していた頃から作っていたみたいですね。
渡部　秀剛(23)滋賀県



▶やり込めばやり込むほどスゴさがわかるスルメゲームではないでしょうか。
石田　智義(21)京都府

▶「PRACTICE」があるから（笑）。これのおかげで，コンティニューなしで１周できました。　三浦　英樹(20)埼玉県

▶コナミの技術力は僕の想像をはるかに超えていた。出たな!!ツインビーでは「敵弾」が消えるなどがあったから，グラディウスIIは無理じゃねえのか？　と思っていたのだが……。ちょっと遅くなるけど，我慢できる程度だから安心した。さらに音楽は，内蔵音源では現在最高レベルだと思う。MIDI(MT-32)を使うと，ドラムとボイスがかぶっても，だいじょう「V」，さらにアレンジもしてある。特にステージ8－1と8－3は原曲を知っていると，違うアレンジで２倍楽しめるぜ！　戸辺　靖(17)栃木県

▶いままでにこんなにいたれりつくせりのゲームがあっただろうか。自分の持っているゲームのなかでドラゴンスピリットと並び気に入っている。今井　裕人(19)茨城県

▶コナミさんへ感謝の意を込めて……。
尾形　淳一(39)秋田県

発売中のソフト

★ウルティマIV　ポニーキャニオン
X68000用　5″2HD版　9,800円(税別)

★三國志III　光栄
X68000用　3.5/5″2HD版　14,800円(税別)

★シュートレンジ　ビッツー
X68000用　3.5/5″2HD版　9,800円(税別)

★バトルテック～失われた聖杯～
ビクター音楽産業
X68000用　5″2HD版　9,800円(税別)

★ヨーロッパ戦線　光栄
X68000用　3.5/5″2HD版　12,800円(税別)

新作情報

★F29 RETALIATOR　イマジニア
X68000用　5″2HD版　価格未定

★メガロマニア　イマジニア
X68000用　5″2HD版　価格未定

★ふしぎの海のナディア　ガイナックス
X68000用　5″2HD版　価格未定

★究極タイガー　金子製作所
X68000用　5″2HD版　価格未定

★TATSUJIN　金子製作所
X68000用　5″2HD版　価格未定

★ファイナルファイト　カプコン
X68000用　5″2HD版　価格未定

★保存版ロードランナー　システムソフト
X68000用　5″2HD版　7,800円(税別)

★ドラゴンスレイヤー英雄伝説　SPS
X68000用　5″2HD版　価格未定

★OVERTAKE（仮）　ズーム
X68000用　5″2HD版　価格未定

★ウェルトリス　BPS
X68000用　5″2HD版　7,800円(税別)

★沈黙の艦隊　ジー・エー・エム
X68000用　3.5/5″2HD版　12,800円(税別)

★エトワールプリンセス　エグザクト
X68000用　5″2HD版　価格未定

★バーンウェルト　グローディア
X68000用　5″2HD版　価格未定

★リーディングカンパニー　光栄
X68000用　3.5/5″2HD版　12,800円(税別)

★ポピュラスII　イマジニア
X68000用　5″2HD版　12,800円(税別)

★キャッスルズ　ビクター音楽産業
X68000用　5″2HD版　9,800円(税別)

★倉庫番リベンジ/ユーザー逆襲編
シンキングラビット
X68000用　5″2HD版　価格未定

★ファルディア　M.N.Mソフトウェア
X68000用　5″2HD版　価格未定

★エアバスター　金子製作所
X68000用　5″2HD版　価格未定

★サバッシュII　ヒッパロスの風
ポプコムソフト/グローディア
X68000用　5″2HD版　12,800円(税別)

★ヴェルスナーグ戦乱　ファミリーソフト
X68000用　5″2HD版　価格未定

新製品紹介

SX-WINDOW対応音色エディタ

SOUND SX-68K

Kioi Makoto 紀尾井 誠

待望のSX-WINDOW用アプリケーションの登場です。まずは，サウンドツール。マルチウィンドウ環境でFM音源の音色をエディットできるようになります。MML派ならすぐにでも実用になるツールです。

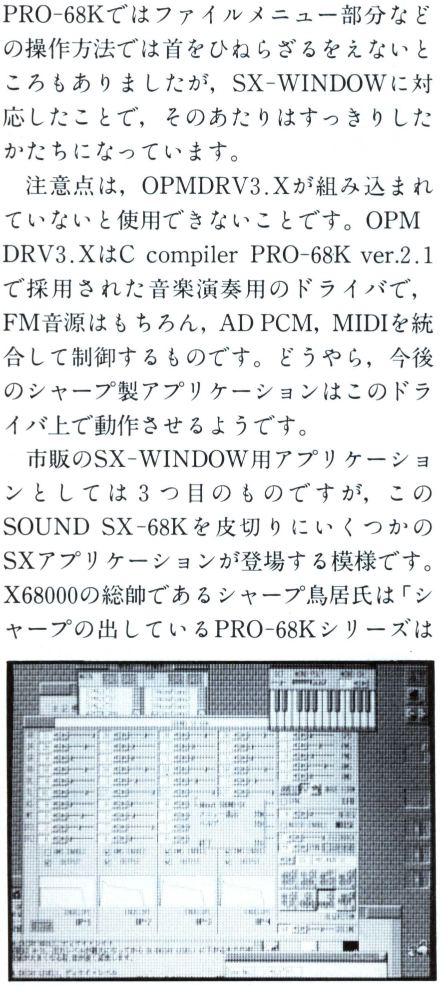
SXアプリケーション登場

SX-WINDOW用のアプリケーションとしてSOUND SX-68Kが登場しました。これは従来のFM音原用音色エディタSOUND PRO-68Kの機能をそっくりSX-WINDOW上に持ってきた感じのものです。

機能的には完全に上位，従来ページ切り替えのような方法で実現されていたものはマルチウィンドウになりました。SOUND PRO-68Kではファイルメニュー部分などの操作方法では首をひねらざるをえないところもありましたが，SX-WINDOWに対応したことで，そのあたりはすっきりしたかたちになっています。

注意点は，OPMDRV3.Xが組み込まれていないと使用できないことです。OPMDRV3.XはC compiler PRO-68K ver.2.1で採用された音楽演奏用のドライバで，FM音源はもちろん，AD PCM，MIDIを統合して制御するものです。どうやら，今後のシャープ製アプリケーションはこのドライバ上で動作させるようです。

市販のSX-WINDOW用アプリケーションとしては3つ目のものですが，このSOUND SX-68Kを皮切りにいくつかのSXアプリケーションが登場する模様です。X68000の総帥であるシャープ鳥居氏は「シャープの出しているPRO-68KシリーズはすべてSX-WINDOWに載せる」といっていたそうですから，今後の展開に注目しましょう。


基本となるメインウィンドウ

機能紹介

マルチウィンドウですからエディタでMMLを記述しながら音色を設定したり，ちょっとファイル操作やほかの作業を……といった場合でもアプリケーションを中断することなく作業できます。将来的には楽譜入力やリアルタイム入力などのソフトとも共存しながら作業が行えるようになるはずです。単なる音色エディタというよりは，統合音楽環境のための最初の製品といったところでしょうか。

基本となる音色パラメータのエディットはメインウィンドウにまとめられ，オプション機能である波形表示などはマルチウィンドウで行われます。

メインウィンドウはOPM（FM音源LSI）の各レジスタ値を設定するオーソドックスなものです。厳密な意味では，こういったツールの使用に際しては各パラメータの働きをはじめFM音源の構造についての知識が不可欠です。よくわからない人向けに，SOUND SX-68Kではヘルプ機能によって，各パラメータの簡単な解説を参照することができるようになっています。また，サンプルの音色が3セット分用意されているので，初心者の方はそれらをいじりながら各パラメータの動作を把握していくのがよいでしょう。

さらに，イメージモードでは「明るい」「柔らかい」「ビブラートのかかった」といった具体的な言葉に対応した基準で操作を行うことができます。

イメージモードのパラメータを操作すると，メインウィンドウのパラメータがリアルタイムに確認できるというのは，SOUND PRO-68Kではできなかったメリットです。必ずしも適切な対応ではない場合もありますが，音作りの初心者にとっては，どのパラメータがどんな役割にあるのかをつかむにはよい機能でしょう。しかし「音の柔らかさ」で無条件にアルゴリズムを変更されたのにはたまげましたが……。

鍵盤表示を行ってマウス操作で，またはキーボードを鍵盤に見立てて演奏することが可能です。注目されるのはPOLYモードの存在です。同時に押されたキーを読み取ってキーボード上から和音を確認することができるのです。同時キー入力に制限のあるX68000のキーボード構成から考えると快挙といえるでしょう。

リアルタイムエディット

SOUND PRO-68Kと同様，サンプル曲を流しながらリアルタイムにエディットの結果を確認できるようになっています。なお，収録されているサンプル曲はSOUND PRO-68K，音色はMUSIC PRO-68Kについてくるものと同じようです。

さらに，サンプル曲だけではなく，サウンド.X（SXPLAY.Xは不可）などで演奏されている曲の音色をいじることができますから，より実践的な内容になったといっていいでしょう。通常はチャンネル固定モードで，音色番号を切り替えていくようになっていますが，音色固定モードにしてサウンド.XでOPMデータを演奏すると演奏中の音色がそのままエディットできるのです。

このモードの優れた点はエディタ画面が単にパラメータをFM音源に送るだけでなく双方向のモニタ&エディタとして機能する点です。つまり，音色のパラメータが変更されるたびにウィンドウ上のパラメータがリアルタイムに変化するのです。サンプルに立川君の曲を聞いていると，ときどき微妙なところがピクリと動く……，うーむ，なるほど（と，よくわからないがとりあえ

ず納得する)。

このモードでは，通常音色番号が指定(表示）されているところがチャンネル番号に変わり，音色のアップダウンと同じ操作でチャンネルを切り替えることができます。音色とFM音源テクニックの研究用には非常に優れた機能といえるでしょう。できれば複数チャンネルの同時モニタ機能（せっかくマルチウィンドウなんだし）や演奏チャンネルのマスク機能がほしかったところです。

操作体系

基本部分の操作方法は各パラメータに対応したスイッチないしはスライダーをマウスでいじるだけのわかりやすい構成になっています。スライダーは独自のものを作成しているようですが，どうせなら縦方向のスライダーのほうがわかりやすかったように思われます（なぜかSX-WINDOWには縦方向のスライダーがない)。

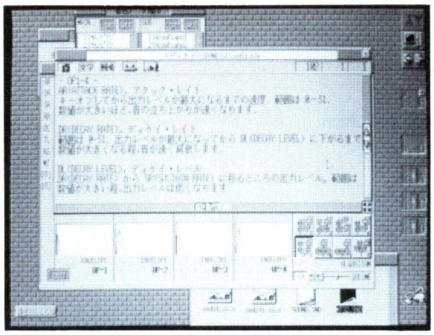
メインメニューほか，そこから呼び出される機能はEasyPaintと同様のサブウィンドウを使用したシステムとなっています。これは前作のEasyPaintでは非常に評判が悪かった操作体系ですが，SX-WINDOW ver.2.0からはアイコンを拾ってくるだけなら，ディレクトリウィンドウをアクティベートしなくなりましたので，ちょっと曲データを放り込む，といった操作ではサブウィンドウの書き換えはまったくなくなりました。

さらに，OPT.1キーを押しながら操作することでアクティベートさせずにウィンドウ操作することができます。もちろん，アプリケーションがそのように対応していなければなりませんが，SX-WINDOW ver.2.0に付属のものであればたいてい大丈夫なようです。逆に，ユーザーが作成したアプリケーションではほぼ全滅です。これも今後はSX-WINDOW上の標準作法として位置づけられていくのでしょう。

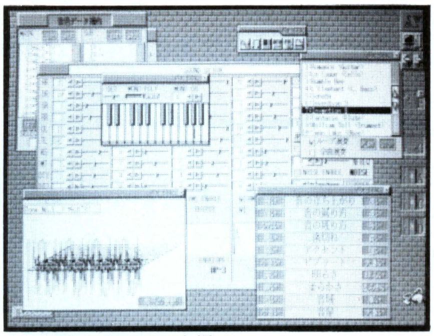
いずれにせよ，これらのシステムのおかげでEasyPaint発表の頃よりはサブウィンドウの扱いにうっとうしさを感じなくてすむようになりました。

しかし，多少操作法に疑問を感じる部分もあります。

いくつかの事例を見ましょう。音色の選択はアップダウンボタンで音色番号を切り換える，直接番号を指定する，音色設定ウィンドウからセットするという3つの方法で行われます。



ヘルプ機能



マルチウィンドウの本領

直接音色番号を指定するのはさておき，通常は音色名を検索していくことが多いと思います。しかし，アップダウンボタンを押すとウィンドウ内のスライドバーなどをすべて書き換えるので少し時間がかかります。連続してボタンを押すとイベントが溜まってしまい，なかなか目的の音にたどりつきません。

音色設定ウィンドウを使えば，SOUND PRO-68Kのように音色の一覧から選択することができます。しかし，ほかのメニュー項目と違って，音色設定だけはサブウィンドウを使用していません。このため，音色設定をしようとすると開いているサブウィンドウをすべて閉じて音色設定後にまたサブウィンドウを復帰するといううっとうしいことになります。

表示が十分に速ければ問題はないのですが，CP/Mの昔から「あまり速くない表示環境でのプログラム作りのノウハウ」は確立されているはずですので，もっと考えてほしかったところです。「書き換え中に次のボタンが押されたら書き換えを中断して次の処理をする」くらいの処理はS-OSのエディタでさえやっていることです。

使用するサブウィンドウは常にメインウィンドウより高い優先順位で表示されますので，場合によっては操作の邪魔になることもあります。サブウィンドウ間にも固定の優先順位があるので通常のマルチウィンドウ操作とは違った感覚で操作しなければなりません。まあ，これによってどんなにウィンドウを開いていてもメインメニューを見失うことはありえませんから，一長一短というところですか。

個人的な感想としては，サブウィンドウの使用がメインメニューだけに抑えられていれば，もっと秀逸な操作体系になったのではと思わされます。

今後の展開に期待

最初にも書いたようにSOUND SX-68KはSX-WINDOW上での音楽ソフト第1弾です。PRO-68KシリーズがSOUND PRO-68K，MUSIC PRO-68K，Sampling PRO-68K，Musicstudio PRO-68Kの順に展開していったように，今後，関連アプリケーションが発売されることが予測されます。

そして，それらが揃ってこそSX-WINDOW上でこのツールを使う真の意味が表れてくるのだと思います。早く第2弾，3弾のソフトを見てみたいものですね。今後の展開に注目しましょう。

SOUND SX-68K	価格未定
シャープ	☎06(621)1221，03(3260)1161

Z-MUSICとの関係

Z-MUSIC上ではSOUND SX-68Kを使うことはできません。ZMUSIC.XとOPMDRV3.Xを切り替えて使用することはどうでしょうか。なにも組み込まない状態でSX-WINDOWを立ち上げ，SX-WINDOW上からドライバを起動することでどちらも組み込みを行うことはできます。あまり気持ちのいいものではありませんがZMUSIC.XもOPMDRV3.Xも常駐解除できるのでこのようなことも可能になります。しかし，PCMDRV.SYSが組み込まれているとOPMDRV3.Xは組み込めないので，この方法をとるとZ-MUSIC常駐時にADPCMが再生できなくなります。完全な同居は不可能なようです（ADDDRVという手はあるが)。

Z-MUSICを使った場合，すでにSX-WINDOW上でもカモンミュージックのRCM形式，X68000の内蔵音源ではもっともデータの蓄積の多いMXDRVやNAGDRVなどのデータがそのまま演奏できる環境が整備されつつあります。Z-MUSIC自体のデータの蓄積もありますので，どちらが標準となるかは自明のことでしょう。

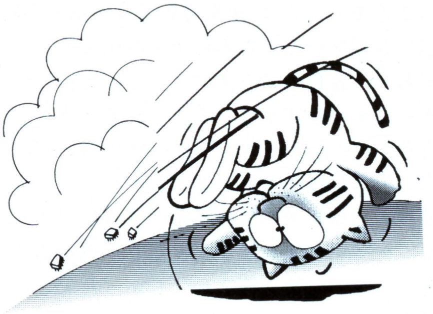
いまのところ，Z-MUSIC用にパッチを当てるか，Z-MUSIC上でSOUND SX-68Kに負けない音色エディタを作ればいいだけの話です。見たところ特に実現困難な機能はありません。波形表示はFB（フィードバック）のかけ方がちょっと面倒ですが，FBに関する限りSOUND SX-68Kの表示も正確ではありませんので，だいたいの雰囲気が出せればそれでよいのでしょう。

問題は次に控えているであろうMUSIC SX-68Kですが，さて……。

吾輩はX68000である
［第15回］

飛び出せ！ ディスク

Izumi Daisuke
泉　大介



吾輩が自慢とする機能のひとつに
オートイジェクト機能がある
今回はこれに焦点を当てよう

グラフィック機能の紹介がひととおり終わったところで，１回の休憩をいただいた。この機に大海を覗き見てみると，その様相があまりに変わってきているのに驚かされる。386SX/DXはいまやMS-DOS系のマシンの標準CPUとなり，とりわけ386SXは入門機に採用されるまでに安直なCPUと化してしまっている。話題の中心は，486SXとそのクロックスピードを倍速化するオーバードライブプロセッサであり，同様のテクノロジーをひとつのCPUに詰め込んだ486DX/2である。クロック25MHzの486SXや486DXが，オーバードライブプロセッサをコプロ用のソケットに装着するだけで（486DXは486DX/2にCPUを差し替えるだけで）50MHzのCPUに化けるというのである。誕生当時にはパソコンの中で比類なき能力を誇っていた我が頭脳MC68000は，いまやはるか後方に取り残されてしまった感を否めない。

もちろん，諸兄もご存じのようにパソコンの能力はCPU性能のみにて決まるものにあらず，グラフィック表示能力と音楽演奏機能は常に吾輩の自慢とするところであったが，そのグラフィック表示能力も昨今の動静を見ていると，いささか心もとない観がなきにしもあらずである。世界ではIBM PCのVGAと呼ばれるボードの表示能力（640×480×16色）に飽き足らず，それを超える能力を持ったスーパーVGAが標準になりつつある。表示できる色数を1677万色中の256色とするのは当然として，解像度も640×480から800×600，あるいは1024×768へと移行する過程にある。512×512ドットで65536色という吾輩の能力では解像度にいささか問題があり，しかもアスペクト比（ドットの縦横比）が１でない。アスペクト比が１となる768×512ドットのモードでは16色しか表示できない。せめて，実画面1024×512ドットで256色（表示画面は768×512ドット）というモードをシャープ大人が吾輩を作るときに考えていてくれればと，いまさらのように思えてならない（あぁ，○○氏の「美しくない！」という言葉が聞こえてきそうである）。

しかしながら，グラフィック表示能力に関しては最近，面白いデータをうちの御仁が見つけてきた。パソコン通信で使われているQ4と呼ばれる拡張子の付いたものである。これは，PC-9801シリーズでグラフィックを楽しむために考えられたフォーマットらしく，16色のグラフィックデータを圧縮したものである。この16色データがふるっているのだ。モノクロ16階調ならそれなりに見られる画像となることは想像に難くない。しかるにこれはカラー画像なのである。カラー画像といえども，アニメ調の絵ならば16色もあればそれなりに見られる絵が作れるのは衆目の認めるところである。しかるにこれは写真などを取り込んだフルカラー（RGB各256階調の1677万色）グラフィックデータをベースにしたものなのである。

通常フルカラーのグラフィックデータを16色にコンバートしたなら，ディザの斑点が画面中ところ狭しと溢れかえるものであり，このQ4フォーマットの画像の中にもそのようなデータは存在する。しかしながら大半のデータは，これが実にうまく抑え込まれているのである。それと指摘されなければ，256色のデータだと思うことだろう。おそらく，もととなったフルカラー画像をQ4データにコンバートするソフトウェアの性能がとんがっているのだ。もちろんパレットが変更されているため，異なる絵を同時に表示すると，先に表示したほうの絵は写真のネガのようになってしまうのはいかんともしがたいところである。このデータを発見して，御仁はいたく感動したようである。「16色というデータ形式もまだまだ捨てたものじゃない」という，そんな心境なのかもしれぬ。吾輩もハードウェアの制約を乗り越えるソフトウェアの勝利には，ただただ頭が下がる思いである。なかには「擬似インターレース」と称し，垂直帰線期間中にパレットを

CompuServeで配布されているLouvreをコンバート

変更することで32色の表示を目指したものまで存在する。ここまで使い切ってもらえれば，コンピュータ冥利に尽きるというものである。

爪に灯をともすような努力がなされている一方で，贅を尽くしたデータも大手を振って罷り通っている。フルカラーデータである。こちらも方向性は異なるもののソフトウェアによる努力はなされており，ファイルサイズを驚くほどコンパクトに圧縮するJPEGなるフォーマットが脚光を浴びている。人間の目のいい加減さに注目したこの圧縮方法は，元絵の完全な再現はできないものの，ファイルサイズを1/10程度にまで圧縮しても元絵の雰囲気をほとんど損なわないという特長をもっている。

次期X68000の仕様はまだ憶測の域を出ないが，卓越したグラフィック機能を持って世に出た吾輩なれば，ここは是非とも後者の勇になりたいものである。互換性を考慮するなら，グラフィックVRAMを２Mバイトに増やし1024×1024ドットで1677万色中の65536色表示というのも悪くない。おっと，こんな安易な展開ではシャープ大人に失礼というものか。失言であった。

◆メモリマップトI/O，再び

これまでに何度かメモリマップトI/Oという言葉を紹介してきた。画面に文字を表示したり，キーボードからデータを受け取ったりという吾輩のデータ入出力機能が，メモリにデータを書き込んだり，メモリからデータを読み出すことによって実現されている，という例のアレである。実例として，メモリにデータを書き込んでいけばそれがすぐさま画面に表示されるという例を，テキスト画面，グラフィック画面を使って紹介した。Z80ではI/Oは独立して扱われており，

　　OUT（アドレス），A

という形式で特定のI/OアドレスにAレジスタのデータを出力し，

　　IN A,（アドレス）

という形式でI/Oからデータを受け取るのが通常である。これに対して吾輩は，メモリにデータを書き込むときに使用できる数々のアドレッシングモード（データ書き込み後，自動的に使ったアドレスレジスタのデータを大きくするポストインクリメントなど）を，I/O操作にもそのまま利用できるという特長をもっている。

画面というI/Oは普段からあまりにも何気なく使っているだけに，I/Oという感じがいまひとつしないかもしれない。そこで今回は，いかにもI/Oという周辺機器を取り上げ，メモリに直結しているところを紹介してみることにしたい。

例として俎上に上げるのはジョイスティックである。吾輩のアプリケーションの中でも最も充実しているゲームソフトを満喫するためにも，ジョイスティックは必須アイテムといえよう。吾輩がやってくるまではとんとゲームに興味を示さなかった御仁が，イのいちばんに買ってくれた周辺機器がジョイスティックであった。あいかわらずゲームは下手で，吾輩が携え，飽きるほどプレイしているはずのグラディウスすらクリアできない腕なのだが，ジョイスティックを購入してからはいっぱしのゲーマー気取りであることは，いつぞやのゲーム特集の折にご紹介したとおりである。

さて，このジョイスティックは図１のような構造をしている。少々わかりづらいかもしれないが，概念図ということで勘弁していただきたい。十文字の足の下に置かれている４つの四角形はスイッチである。ここでスティックを上に倒すと(1)のスイッチが入り，スティックを左に倒せば(3)のスイッチが入る。そして左上に倒したなら，(1)と(3)のスイッチが同時に入るという仕組みである。これで，どのスイッチが入っているかを調べればジョイスティックがどの方向に倒されているのかを知ることが可能となる。

スイッチが入ったかどうかは，コンピュータの常によって１と０で表現できる。つまり，４桁の２進数があれば，ジョイスティックの状態は表現できるというわけである。実際にはこれにA・Bの２つのボタンの状態を表すデータが加わり，６桁の２進数によってジョイスティックの状態が表現されている。そしてこのデータが割り付けられているのが，E9A001$_H$とE9A003$_H$の２つのアドレスである。

E9A001$_H$とE9A003$_H$のデータは図２のようになっている。それぞれのスイッチの状態が，６つのビットを使って表現されているのが確認いただけるだろう。ではデバッガを使って，実際にデータを確かめてみよう。メモリのダンプを行うDコマンドを使って，E9A001$_H$のデータを表示してみたのが図３である。注意していただきたいのは，ジョイスティックをただつないだだけの状態では，図３-１のように全スイッチのデータが１となっている点である。これは，ジョイスティックのスイッチには常に5Vの電圧がかかっていることを意味している。スティックが倒されたりトリガボタンが押されてスイッチが入

図１　ジョイスティックの動作原理

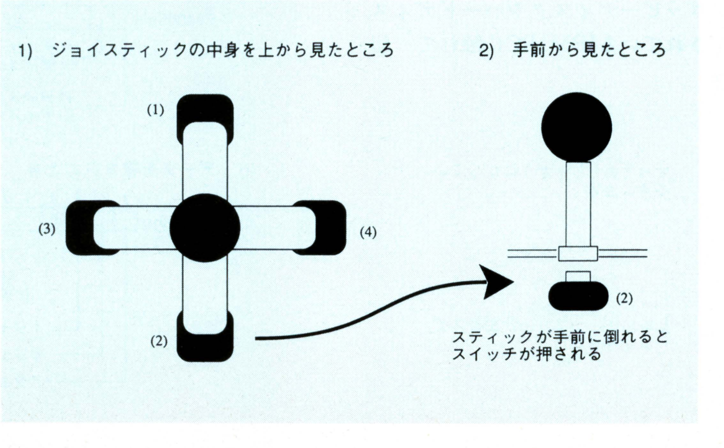
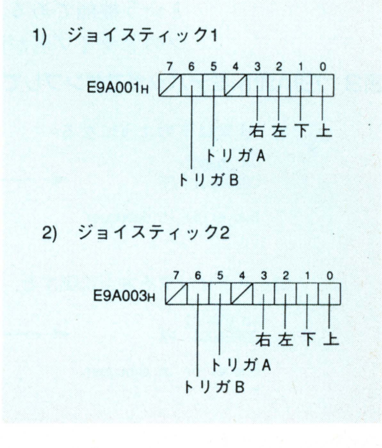


図２　ジョイスティックデータの読み取り



ると，アースされて電圧は0Vに落ちる。つまり，該当するスイッチのデータは0となるのである。たとえば，スティックを左上に倒したままでDコマンドを実行すると，図3-2のように上を表すスイッチと左を表すスイッチが同時に0となる。スティックをさまざまな方向に倒したり，トリガボタンを押して実験してみていただきたい。このとき，データが1バイトだけ表示されてバスエラーとなるのは，E9A002$_H$からデータを読むことができないためである。気にせず実験していただきたい。

◆フロッピーディスクを吐き出せない

吾輩はオートローディング/オートイジェクトのフロッピーディスクドライブを備えている。データの読み書きヘッドを下ろすための武骨なハンドルをドライブの前面から駆逐した洗練されたデザインは，吾輩の常に自慢とするところである。これに影響されたのか，DOSマシンの中にもハンドルを持たないものが登場してきている。見た目は吾輩のディスクドライブ同様，ディスクを挿入するスロットとボタンがあるだけというシンプルな構造なのだが，騙されてはいけない。ヘッドを自分で下ろさなければならない構造は相変わらずで，ディスクを挿入したあと，あろうことかこのボタンを押す必要があるのである。ハンドルをボタンに変えただけのこの構造になにか意味があるというのか。理解に苦しむところである。

御仁は吾輩のオートイジェクト方式のディスクドライブが，たいそうお気に入りらしい。ゲームソフトの中には必要なプログラムやデータを読み込んでしまったらディスクを自動的に吐き出すものや，ディスクを入れ替える必要のあるときにはディスクを吐き出すものが存在するが，このあたりにX68000の美学を感じているようである。いったんこの類のソフトに触れてしまうと，そうでないものはことごとく手抜きに感じられるらしい。悪態をつくこともしばしばである。なかでも，ディスクを吐き出す場合とそうでない場合が混在している「遙かなるオ○ガスタ」は，そのアンチユーザーフレンドリなメニューシステムともあいまって恨みを買っている。

吾輩のオートイジェクト機能の中でも御仁が特に気に入っているのが，CTRLキーを押しながらF1，F2キーを押せば，ドライブA，Bのディスクが吐き出される（A，Bがハードディスクの場合はヘッドがリトラクトする）という機能である。フロッピーディスクやハードディスクのドライブ名が固定されているIBM PCに触れて，起

図3　E9A001$_H$をデバッガでダンプしてみる

1)　通常は下のようになる

```
-d e9a001
00E9A001  FF
```

←　すべてのビットが1になっている点に注意

```
bus error in debugger
-
```

2)　スティックを左上に倒すと

```
-d e9a001
00E9A001  FA
```

←　A_H=1010$_B$。つまり上と左のスイッチが押された

```
bus error in debugger
-
```

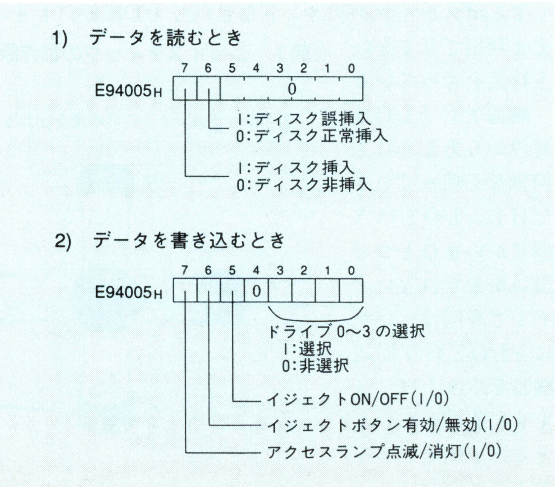
動するメディアによってドライブ名が変更されてしまうHuman68kもかくあるべしとばかりに「フロッピーディスクはA，Bに固定しよう」という原稿を書き上げスタッフの顰蹙を買ったことがあるが，実はこれはハードディスクから起動した場合にも，CTRL＋F1，F2でフロッピーディスクを吐き出したいがための方便であったことは吾輩のみが知る事実である。このときに御仁が作ったプログラムがrendrv.xであった。これはDRIVE.Xコマンドのように2つのドライブ名を交換するのではなく，指定したとおりにドライブ名を一斉に付け変えてしまうというコマンドである。不精な御仁にしては，なかなか気のきいたプログラムだと，吾輩も感心した経過がある。

そんな御仁が，フロッピーディスクをコントロールし吐き出すためのI/Oもメモリにマップされているという吾輩の特長を知って真っ先に試みた実験がある。デバッガを使ってフロッピーディスクを吐き出させようというのである。フロッピーディスクのイジェクトはIOCSコールに用意されているため，これを使うプログラムをデバッガでチョチョイと作成すれば実に簡単に実現できる。そうではなく，御仁がやりたいと思っていたのは，ディスクドライブのI/Oがマップされているメモリを直接操作してフロッピーディスクを吐き出させようという実験である。

吾輩のメモリマップを見て，フロッピーディスク操作を行うためのI/Oが図4のように割り当てられていることを知った御仁は，E94005$_H$にさっそく00100001$_B$というデータを書き込んでみた。つまり，ドライブ0に挿入されているディスクを吐き出せという指示である。ガウォンという音とともに吾輩がディスクを吐き出すのを期待して待ち受けた御仁だったが，結果はものの見事に失敗に終わった。御仁はフロッピーディスクをコントロールしている吾輩の回路が，どのような「タイミング」でこのデータを受け取るように設計されているのかを知らなかったのである。

これは実にハードウェア的なタイミングなので，簡単な例を挙げて説明しよう。吾輩の基幹をなしているのはデジタル回路だが，デジタル回路を構成する部品の中で最も基本的なものとしてNAND（ナンド）ゲートと呼ば

図4　フロッピーディスクコントロール



れるものがある。NANDゲートはNot ANDを略したもので，２つの入力がどちらも１の場合だけ０を，それ以外の入力には１を出力するという特徴がある。このNANDゲートを４つ使うと，D-フリップフロップと呼ばれる図５のような回路ができる。フリップフロップはデータを保持する役目を持った回路で，吾輩の中でもそこここに使用されている回路である。このD-フリップフロップは，図５のIN 1に与えられたデータと同じものをOUT 1に，その反対のデータをOUT 2に出力する。そして，いったんデータが保持されたら，IN 2が０であるかぎり出力されるデータは変わらず保持され続けるという特長を持っている。つまり，IN 2が０になっているときに，IN 1をいくら変更してもOUT 1/2は変わらないということだ[1)]。

こんな能書きばかりでもつまらないので，簡単な実験プログラムを用意した。リスト１である。これはD-フリップフロップの動作をX-BASICでシミュレートしてみようというもので，実行すると画面に，

in 1	in 2			out 1	out 2
0	0	1	1	1	0

という形式でデータが表示されるようになっている。最初の２つの入力データに続くデータは，図５の４つのNANDが出力するデータを表しており，順に左上，左下，右上，右下のNANDに対応している。「１」キーでin 1を，「２」キーでin 2を変更できるようにしてあるので試してみていただきたい。in 2が０のときに，in 1をどんなに変更しても出力データに変化がないことを確認していただけるだろう。out 1を０に（そしてout 2を１に）するためには，

1)　in 1を０にする
2)　in 2を１にする

という手順を踏まなければならない。この手順をよーく味わっていただきたい。御仁がフロッピーディスクを吐き出させることができなかった理由も同様のものなのだ。

---

1)　ここでは便宜上IN 1，IN 2，OUT 1，OUT 2という用語を用いたが，一般にはD，CLK，Q，$\bar{Q}$と表記される。

◆フロッピーディスクを吐き出すには

D-フリップフロップでin 1に与えたデータを出力に反映させるためには，in 2を1にしなければならなかったが，これを逆にいうとin 2が1になってさえいればデータは素通りするということである。吾輩のように高級なパソコンでは，このようにいい加減なことでは危険きわまりない。そこで採用されているのが，in 2が0から1になる瞬間（in 2の立ち上がり），あるいは1から0になる瞬間（in 2の立ち下がり）にデータを取り込むというロジックである。フロッピーディスク吐き出し部分もこのロジックでできており，御仁の方法は手順の途中までしか実行していなかったのである。正しくは次の順にデータを出力しなければならない。

1)　00100001_Bを出力する
2)　00100000_Bを出力する

つまり，ディスクを吐き出させたいドライブを指定するビットを，１にしたあと０に戻す手順が必要なのである。２つの出力データで指定された動作が同じ動作であった場合にのみ，吾輩のディスクドライブは指定されたとおりの振る舞いを見せる（この場合はディスクのイジェクト）。

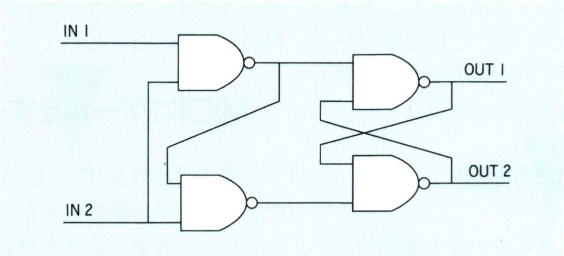
アドレス$E94005_H$で指定するさまざまな動作は，すべてこの手順で機能するようになっている。遊んでみていただきたい。なお，奇数アドレスに１バイトのデータを書き込むわけなので，デバッガのMEコマンドに１バイト単位の書き換えを指示するＳをつけて，

-mes e94005

として作業しなければならないのをお忘れなく。

吾輩につながれているI/Oは，すべてこのようにメモリ操作によって動作の指示が行えるようになっている。では，キーボードは，マウスはどうなっているのかと疑問に思われることだろう。残念ながら，これらのデバイスとのやりとりはそれほど簡単ではない。MZの頃にはキーボードはI/Oに直結されており，ジョイスティックと同様の方法で，どのキーが押されたのかを調べることができるようになっていたが，吾輩はX1同様にキーボードを別のCPUに管理させているのである。キーが押されるとこのCPUは吾輩に割り込みをかけてくる。そして，押されたキーのデータを通信を介して吾輩に転送するのだ。マウスも同様である。このため，生のデータに触れるのはなかなかに難しい。これは機会があれば，ということにしておきたい。

図5　D-フリップフロップの回路図



リスト１

```
 10 /*
 20 /*     Dフリップフロップの実験
 30 /*
 40 int gate(5)
 50 int i
 60 str ky
 70 /*
 80 cls
 90 locate 16, 14
100 print "in 1","in 2",,,"out 1","out 2"
110 while 1
120   ky = inkey$(0)
130   if ky = chr$(27) then break                /* ESCキー
140   if ky = "1" then gate(0) = not gate(0)
150   if ky = "2" then gate(1) = not gate(1)
160   /*
170   gate(2) = nand( gate(0), gate(1) )
180   gate(3) = nand( gate(2), gate(1) )
190   gate(4) = nand( gate(2), gate(5) )
200   gate(5) = nand( gate(4), gate(3) )
210   /*
220   locate 16, 16
230   for i=0 to 5
240     print abs( gate(i) ),
250   next
260 endwhile
270 end
280 /*
290 func nand(x, y)
300   return( not ( x and y ))
310 endfunc
```

新製品紹介

GM対応音源モジュールTG100

Takahashi Tetsushi 高橋 哲史

楽器の共通規格であるGM対応の音源モジュールとして，日本楽器からTG100が発売されました。最大28声同時発音で45,000円という低価格が魅力です。加えて既存のMIDIデータへの対応モードなども見逃せません。

さて私こと高橋哲史はSION IIのBGMですっかり曲作りの魅力にとりつかれてしまいました（もう，どっぷり漬かってます)。そこでこれを機に，脆弱なる我がMIDIシステムをバージョンアップしてやろうと思い，新音源の購入を画策したのです。なにがいいかなぁ，やっぱりSC-55，いやSC-155かなぁ，でも思い切ってM1Rとか買っちゃうのもいいなあ……などと思いをめぐらしていたとき，私は編集部でU氏の囁きを聞いてしまったのです。

「今度ヤマハから出る TG100 って安い割にかなりいい音を出すみたいだね。GM対応でDTMにも適してそうだし」

翌日私の部屋にTG100が鎮座していました。そう，私は勢いと欲望のみに生きる男（笑)。でも転んでもただでは起きないっ，ということでヤマハから発売された新音源TG100を紹介させていただきます。

TG100の概要

この上の文章だけ見るとかなり衝動的に購入したかのように思われてしまいそうですが，実際は１日潰してあちこち歩き回り，いろいろな音源と比較して最終的にTG100を選んだのです。私もこんな大きな買い物を衝動買いで済ませられるほど裕福な経済状態じゃないですからね。

TG100はGENERAL MIDI対応（レベル1)のマルチティンバー音源です。コンパクトなハーフラックサイズながら，AWM音源28ポリフォニック，202種類の音色に10種類のドラムキット8種類のリバーブ，RS-232Cを介してのコンピュータとの接続，AUDIO IN端子による簡単なミキシングなどなかなか力の入った仕上がりになっています（入力レベルを調整できます)。

また3種類のサウンドモジュールモードを備えており，既存の音楽データやヤマハディスクオーケストラコレクションなどの利用も可能になっています。

MIDIでゲームをする人には

まず，ずばりMIDI対応のゲームをしたい，あるいは市販の（または通信などで流れている）音楽データを聴いてみたいという方々にとってこのTG100はどうでしょうか？　結論から先にいうと，TG100はそういったニーズにもかなり応える仕様になっているといえるでしょう。。

TG100 45,000円(税別)

既存のデータのほとんどはMT-32やCM-64,最近ではSC-55などのGS音源で作られています（DTMの世界ではやっぱり圧倒的にローランドが強いですね)。TG100はそれらのデータをすべて利用できるように設計されているのです。つまりMT-32やCM-64などで作られたデータはC/Mモード，SC-55などで作られたデータはGMシステムレベル1モードにすればわりと忠実に再現されるようになっているのです。

とはいってもやはり完全に曲が演奏されるわけではありません。具体的にどのあたりがサポートされてないかを挙げてみましょう。

まずC/MモードによるMT-32,CM-64データの利用ですが，MT-32,CM-64プリセットの音色を使っているものはかなり違和感なく聴けるようになっています。が，プリセットではなくオリジナルの音色が使用されている曲は正常に演奏されません。またCM-64の別売り音色カードを使用しているものについてもサポートされていません(カード部分の音は鳴らない)。また音色の特性の違いにより，ボリュームバランスが若干違って聴こえる場合があります。さらにCM-64のデータなどの場合，さすがにパーシャル（基本的な音数。ヤマハではエレメントと呼んでいるが）が足りずに発音が追いつかないことがあります。ただTG100のほとんどの音色が1パーシャル（＝エレメント）で作られているので，それほどひどい状況になることは少ないようです。

次にGMシステムレベル1モードによるSC-55のデータ利用です。これもかなり聴けるようになっているのですが，やはり少し苦しいところもあります。というのも，GSフォーマットというものはGM規格を内包している，いわば上位互換フォーマットだからです。というより，GSフォーマットはまだローランドが提唱しているだけでJMSC-MMA（MIDI規格協議会，MIDI Manufactuarers Association,USA）などに承認されているわけではなく，いわばローランドローカルな規格ともいえるのです。

GM規格がサウンドセットやパーカッシ

ョンマップ，ボリューム，パンなどデータの互換性が期待される必要最低限の規定しかしていないのに対し，GSフォーマットではさらにつっこんで内蔵エフェクタの操作，NRPNによる音色モディファイなどが細かく規定されています。つまり，それらのMIDIメッセージはGM規格の音源（＝TG100）では，うまく演奏できないのです。

とはいっても聴くだけならばそれほど問題ないといってもよいと思います。もともと音色もかなり充実していますし（個人的にピアノとストリングが好き），本来GM規格では未定義のドラムパートのバンクチェンジもローランドライクなものがサポートされています（SC-55を意識してるんですね）。手持ちのデータ（全部で1000曲くらいかな）やゲームなどを起動して聴いてみましたが，極端にひどい演奏をするものはありませんでした。まあ，7割方は大丈夫，というところでしょうか？　（ただ，ダメなやつはとことんダメなんですけど……。MTの液晶表示やリバーブなどを頻繁にいじるのなんかは最低）。

C/MモードをSC-55のMTモードと比べると基本的な音色ではMTモードのほうが忠実ですが，曲によってはC/Mモードのほうが自然な演奏をすることもあります。C/Mモードにはタイプ１とタイプ２の音色セットがありますが，タイプ１はほぼMT-32，タイプ２はCM-32Pとほぼ同じ音色配列になっていますのでCM-64対応のデータにも対応できるのは強みでしょうか。

ただし，完全にMT-32やCM-64，SC-55のデータが再現されると思っているとがっかりしますので，そのへんは認識しておいてください。SION IIのSC-55版オープニングなどでは，ドラムキットを２基設定していたりといった小技を使っているのでうまく再生できません。凝った曲ほど再現率が悪くなるのは悲しいことです。

自作派の人たちには

次に自分で曲を作って演奏したい，という方々を考えてみたいと思います。おそらく購入の際，SC-55と天秤にかける人が多くおられると思いますので，そのあたりを検討してみましょう。

最初にSC-55に比べて優っている点を考えます。まずなんといっても価格の安さが挙げられるでしょう。SC-55とほぼ同等の性能を備えながら45,000円と，かなり安く抑えられています。私も最終的にはこの低価格の魅力が購入の決め手となりました。

次にRS-232Cを用いたパソコンとの接続が可能だという点が挙げられます。これによりMIDIボードを買わなくても，音源とケーブルさえ揃えれば，すぐにMIDI環境を手中にすることができます。もちろんRS-232Cに対応したシーケンスソフトがなければ意味がないのですが……（PC-9801，Macintoshなどにはあるようですが，残念ながらX68000にはありません）。

そして最後に同時発音数28とSC-55の24音に比べて若干余裕があるところです。たかだか4音，と思われるかもしれませんが，GM/GS音源ではこれが案外大きな差につながります（GM規格で1音色は2パーシャル以下で構成することが決められているので，ほぼパーシャル数＝実発音数）。

それでは次にSC-55に比べて若干見劣りのする点を挙げてみましょう。まず最初に出てくるのは音色エディットの弱さです。演奏専用機器として使うならあまり関係ありませんが，楽器として使うことも考えている人にはちょっと気になるところです。

TG100では64音色分のユーザー領域を確保してあり，そこにプリセットの音色をコピーしてエディットできるようになっています。が，そこでエディットできるのはパーシャルごとの音量，パン，デチューンのみなのです。これに加えて演奏時にチャンネルごとのアタックレイト，リリースレイトも変更できるようになっています。しかしさらにレゾナンスの操作などができるSC-55には，音色の表情の変化においては劣っているといわざるをえないでしょう。

次に挙げられるのは前面パネルでの操作性の悪さでしょうか。6つのボタンと1行の液晶表示だけですべての操作を行うため，ページの切り替えなどを繁雑に行わねばな

表1

音源方式		AWM音源 リバーブ内蔵 最大同時28音発音，後着優先 最大16音色同時発音
マルチティンバー		16チャンネル DVA 1ボイスAWM×２までのレイヤー可能 ドラムトラックの優先発音
音源機能		GMシステムレベル1規格準拠 MIDI BANK SELECTにより音色バンク変更
インタフェイス機能		従来のMIDIシーケンサ，キーボードも接続可能
互換性		GMに対応 ディスクオーケストラに対応 従来のC/Mに対応（一部エクスクルーシブメッセージは除く）
プリセット音色数		202音色（ノーマルボイス192音色，ドラムボイス10音色）
インターナル音色数		64音色
接続端子	フロント	PHONES×1（ステレオミニジャック） AUDIO IN×1（ステレオミニジャック）
	リア	LINE OUT×２(R,L/MONO) MIDI IN MIDI OUT MIDI THRU TO HOST(MINI DIN 8P)
電源電圧		15V500mA
外形寸法		220(W)×196.5(D)×40.6(H)
重量		1.0kg
付属品		電源アダプターPA-1505　×1 取扱説明書　×1
オプション（別売品）		ラックマウントキットRK101

TG100の3つのモード

TG100には３つのモードがあります。それは，GM（General MIDI）音源としてのモード，ディスクオーケストラのモード，そしてコンピュータミュージックモードです。

GMモードは世界的に標準となっている音楽データを演奏するためのモードです。TG100ではSC-55と同様にパワーセットなどといったドラムキットのバリエーションが拡張されており，GS対応の曲データでもある程度対応できるようになっているようです（一部違うものもありますが）。

ディスクオーケストラというのは，従来ヤマハがエレクトーンなど用に出していた音楽データ集を聞くためのモードだそうです。これについては今回はよくわかりませんでした。

コンピュータミュージックモードはローランドCM-64対応のデータを鳴らすためのものと考えてよいでしょう。C/M TYPE1と呼ばれるものがMT-32，C/M TYPE2と呼ばれるものがCM-32Pの音色に対応しています。

単にGM音源というだけでなく，まさに，ありとあらゆるデータに対応しているのです。

らず，慣れないと大変です(私はQY10でもう慣れましたが)。これもリモコンまでがついてきていたれりつくせりのSC-55に比べると辛いところです（パラメータは全部エクスクルーシブで送信すればいいんだし，DTM音源として使うなら問題ないとは思うんですけどね。なにもついていないCM-64やCM-300がDTMの標準なんだし)。

そして細かいところではパンが15段階になっている，GM規格外でGSフォーマットにはある一部のパーカッション音がない（たとえばStandard SetのSlapとか）などの相違点が挙げられます。

以上のような感じなのですが，最終的にはやはり自分の耳で聴いてみてどちらを購入するか，というところでしょう。

最後に

TG100はパソコンユーザーに手が届く安価な価格帯に登場したGM音源といえます。GM規格自体これからどうなるかわからないところがありますが（いまだにレベル1の規定しか存在しないし……レベル2は制定されるのか？　そのときGSフォーマットはどうなる？)，音源の共通化によって得られるメリットについては大いに興味があるところです。これからの動向を見守っていきたいですね。とりあえず私はTG100で目一杯遊ばせていただきます。

表2

General MIDI System Level 1

Prg#	Voice	Prg#	Voice
1	GrandPno	65	Sprno Sax
2	BritePno	66	Alto Sax
3	El.Grand	67	TenorSax
4	HnkyTonk	68	Bari Sax
5	ElPiano1	69	Oboe
6	ElPiano2	70	EnglHorn
7	Harpsich	71	Bassoon
8	Clavinet	72	Clarinet
9	Celesta	73	Piccolo
10	Glocken	74	Flute
11	MusicBox	75	Recorder
12	Vibes	76	PanFlute
13	Marimba	77	Bottle
14	Xylophon	78	Shakuchi
15	TubulBel	79	Whistle
16	Dulcimer	80	Ocarina
17	DrawOrgn	81	SquareLd
18	PercOrgn	82	Saw Ld
19	RockOrgn	83	CaliopLd
20	ChrcOrgn	84	Chiff Ld
21	ReedOrgn	85	CharanLd
22	Acordion	86	Voice Ld
23	Harmnica	87	Fifth Ld
24	TangoAcd	88	Bass &Ld
25	NylonGtr	89	NewAgePd
26	SteelGtr	90	Warm Pd
27	Jazz Gtr	91	PolySyPd
28	CleanGtr	92	Choir Pd
29	Mute Gtr	93	Bowed Pd
30	Ovrdrive	94	Metal Pd
31	Distortd	95	Halo Pd
32	Harmnics	96	Sweep Pd
33	WoodBass	97	Rain
34	FngrBass	98	SoundTrk
35	PickBass	99	Crystal
36	Fretless	100	Atmosphr
37	SlapBas1	101	Bright
38	SlapBas2	102	Goblin
39	SynBass1	103	Echoes
40	SynBass2	104	SciFi
41	Violin	105	Sitar
42	Viola	106	Banjo
43	Cello	107	Shamisen
44	Contra	108	Koto
45	TremStrg	109	Kalimba
46	Pizzicto	110	Bagpipe
47	Harp	111	Fiddle
48	Timpani	112	Shanai
49	Ensmble1	113	TnklBell
50	Ensmble2	114	Agogo
51	SynStrg1	115	Stl Drum
52	SynStrg2	116	WoodBlok
53	AahChoir	117	TaikoDrm
54	OohChoir	118	MelodTom
55	SynChoir	119	SynthTom
56	Orch Hit	120	RevCymbl
57	Trumpet	121	FretNoiz
58	Trombone	122	BrthNoiz
59	Tuba	123	Seashore
60	MuteTrum	124	Tweet
61	FrenchHr	125	Telphone
62	BrasSect	126	Helicptr
63	SynBras1	127	Applause
64	SynBras2	128	Gunshot

Disk Orchestra

Prg#	Voice	Prg#	Voice
1	BrasSect	65	PipeOrgn
2	Trumpet	66	JazzOrgn
3	FrenchHr	67	SynBras3
4	Sax 1	68	Sax 1
5	Clarinet	69	Clavnova
6	Oboe	70	CleanGtr
7	Flute	71	Mute Gtr
8	Acordion	72	WoodBass
9	Ensmble4	73	Jazz Gtr
10	Violin	74	PopBrass
11	PipeOrgn	75	Ensmble2
12	JazzOrgn	76	Violin
13	GrndPno2	77	ChrcOrgn
14	ElPnoDX	78	Sax 2
15	Harpsich	79	Hvy Bass
16	Celesta2	80	Flute 2
17	Vibes	81	Bassoon
18	Marimba	82	
19	Clavinet	83	JazzOrgn
20	Glocken	84	
21	SynBras3	85	
22		86	
23	SynCrstl	87	
24	Timpani2	88	
25	NylonGtr	89	
26	Jazz Gtr	90	
27	CleanGtr	91	
28	Sitar	92	
29	WoodBass	93	
30	FngrBass	94	
31	SlpBas10	95	
32	SynBass2	96	
33		97	
34		98	
35		99	
36		100	
37		101	
38		102	
39		103	
40		104	
41	MuteTrum	105	
42	Harmnica	106	
43	AahChoir	107	
44	CombOrgn	108	
45	Syn Wood	109	
46	SynStrg3	110	
47	SynChor2	111	
48	BritePn2	112	
49	GrndPno2	113	
50	HnkyTonk	114	
51	ElPiano1	115	
52	El.Grand	116	
53	SynPiano	117	
54	SteelGtr	118	
55	CleanGtr	119	
56	Banjo	120	
57	Pizzicto	121	
58	Harp	122	
59	Stl Drum	123	
60		124	
61	BrasSect	125	
62	Flute	126	
63	Ensmble1	127	
64	AahChoir	128	

C/M Type 1

Prg#	Voice	Prg#	Voice
1	GrandPno	65	WoodBass
2	BritePno	66	WoodBass
3	Elpiano2	67	FngrBass
4	El.Grand	68	PickBass
5	El.Grand	69	SlapBas1
6	ElPiano2	70	SlapBas2
7	ElPiano1	71	Fretless
8	HnkyTonk	72	Fretless
9	DrawOrgn	73	Flute
10	PercOrgn	74	Flute
11	PercOrgn	75	Piccolo
12	RockOrgn	76	Piccolo
13	ChrcOrgn	77	Recorder
14	ReedOrgn	78	PanFlute
15	ChrcOrgn	79	Sprno Sax
16	Acordion	80	Alto Sax
17	Harpsich	81	TenorSax
18	Harpsich	82	Bari Sax
19	Harpsich	83	Clarinet
20	Clavinet	84	Clarinet
21	Clavinet	85	Oboe
22	Clavinet	86	EnglHorn
23	Celesta	87	Bassoon
24	Celesta	88	Harmnica
25	SynBras1	89	Trumpet
26	SynBras2	90	Trumpet
27	SynBras1	91	Trombone
28	SynBras2	92	Trombone
29	SynBass1	93	FrenchHr
30	SynBass2	94	FrenchHr
31	SynBass1	95	Tuba
32	SynBass2	96	BrasSect
33	NewAgePd	97	BrasSect
34	SynHarmo	98	Vibes
35	Choir Pd	99	Vibes
36	Bowed Pd	100	MalletSy
37	SoundTrk	101	MaletWin
38	Atmosphr	102	Glocken
39	SynWarm	103	TubulBel
40	SynFunny	104	Xylophon
41	SynEcho1	105	Marimba
42	Rain	106	Koto
43	SynOboe	107	Sho
44	SynEcho2	108	Shakuchi
45	SynSolo	109	Whistle
46	SynRdOrg	110	Whistle
47	SynBell	111	Bottle
48	SquareLd	112	Breathy
49	Ensmble1	113	Timpani
50	Ensmble2	114	MelodTom
51	SynStrg1	115	DeepSnar
52	Pizzicto	116	SynthTom
53	Violin	117	Syn Tom2
54	Viola	118	TaikoDrm
55	Cello	119	TaikoRim
56	Cello	120	Cymbal
57	Contra	121	Castanet
58	Harp	122	Triangle
59	Harp	123	Orch Hit
60	NylonGtr	124	Telphone
61	SteelGtr	125	Bird
62	Jazz Gtr	126	Jam
63	CleanGtr	127	EfcWatr
64	Sitar	128	EfctJngl

C/M Type 2

Prg#	Voice
1	GrandPno
2	GrandPno
3	El.Grand
4	Hnky Tonk
5	GrandPno
6	BritePno
7	BritePno
8	ElPiano1
9	ElPiano1
10	ElPiano1
11	SteelGtr
12	SteeGtr
13	SteelGt2
14	Mute Gt2
15	Mute Gt3
16	SlapBas3
17	SlapBas4
18	SlapBas5
19	SlapBas6
20	SlapBas7
21	SlapBas8
22	SlapBas5
23	SlapBas9
24	FngrBass
25	FngrBas2
26	PickBass
27	PickBas2
28	Fretless
29	WoodBass
30	AahChor2
31	AahChoir
32	AahChor3
33	AahChor4
34	Ensmble2
35	Ensmble1
36	Ensmble3
37	Ensmble3
38	DrawOrgn
39	DrawOrgn
40	PercOrgn
41	PrcOrgn2
42	DrawOrgn
43	DrawOrgn
44	PercOrgn
45	PrcOrgn2
46	PrcOrgn2
47	Trumpet
48	Trumpet
49	Trombone
50	Trombone
51	Trombone
52	PopBrass
53	BrasSec3
54	BrasSec2
55	SprnoSax
56	Alto Sax
57	TenorSax
58	Bari Sax
59	BrasSect
60	PopBrass
61	BrasSec2
62	BrasSec2
63	BrasSec2
64	Orch Hit

ハードウェア工作入門〔26〕

コンピュータアーキテクチャ編

論理演算で加算器を作る

Misawa Kazuhiko
三沢 和彦

今回は，ALUの基本動作である加算器の設計を基礎からみっちり解説していきます。ここで基本となるのは論理演算。すでにお馴染みのAND,OR,NOTを使って，デジタル論理回路を構成していきます。

加算器

先月の導入部でいったとおり，さっそくCPUの動作から学んでいきたいと思います。CPUの基本的な機能といえば，外部から入力したデータを演算することです。もともとコンピュータは，数の計算しかできないといっても過言ではありません。CPUの動作とはすなわち，データの入力端子と出力端子のある電子回路にどのような形でデータを入力すれば，どのように演算結果が出力されてくるのかということにほかならないのです。

CPUは大きく分けて，演算部と制御部とからなっています(図1)。実際に演算を行う部分が演算部で，特に基本的な算術論理演算のできる演算装置のことをALU(Arithmetic and Logical Unit) といいます。そしてもうひとつ，制御部があります。いま扱おうとしているノイマン型コンピュータ(プログラム記憶方式）では，演算命令が加算であるのか乗算であるのかという命令内容自体もメモリに順番に記憶されています。制御部はこのメモリに格納されている命令データを解読し，その命令を実際に行うために制御信号を演算部へ送る役割を果たすのです。実際にCPUの動作を理解するためには，演算を行う中心部であるALU（算術論理演算部）の動作を理解することが大切です。

さて，そもそも演算とはなんでしょうか？ 演算という手続きを電子回路で実行するためにはどのようなハードウェアが必要なのでしょうか？ 今月はそのあたりの基礎的なところから入っていきたいと思います。

演算とは

算数の時間に出てくる演算といえば，四則演算，すなわち，加算，減算，乗算，除算の4種類の演算です。実際この四則演算さえできれば，たいていの計算はこなせます。このほかに，三角関数などの特別な関数計算などを実行する必要も出てきますが，コンピュータの場合は三角関数なども上の四則計算の組み合わせに置き直して計算させることが多いものです。X68000シリーズのMPU68000で実行できる（算術）演算も四則演算のみです。

しかしながら，ALUの中でこの4種類の演算を別々に行っているわけではありません。要するに，この4種類の演算はすべて独立かというとそういうわけでもないのです。たとえば減算を考えてみましょう。減算においては,「引く数を負の数に置き換えて加えてやる」と考えると，これも加算の一種になるのです。具体例を挙げると，

$$6-2=4$$

は6から2を引く減算です。ここで引く数の2を負の数−2に置き換えて，

$$6+(-2)=4$$

図1 CPUの内部構成

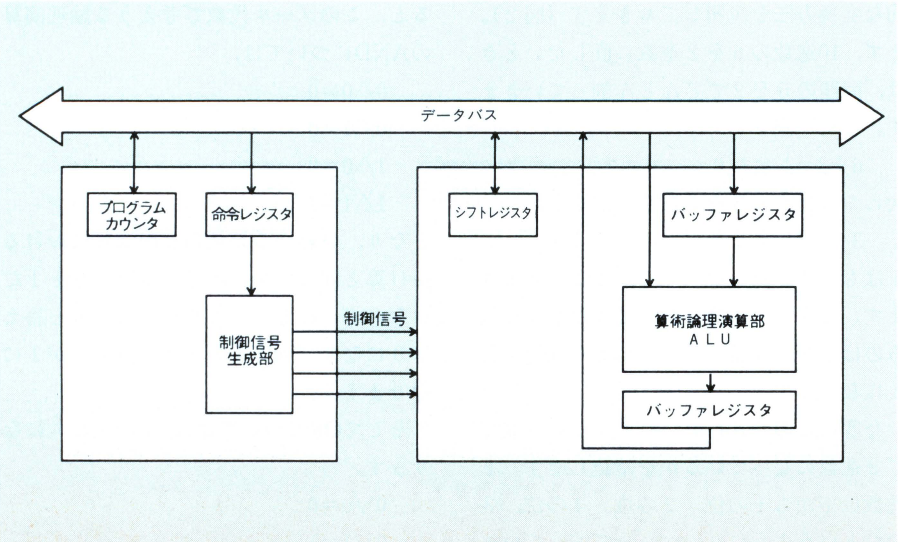
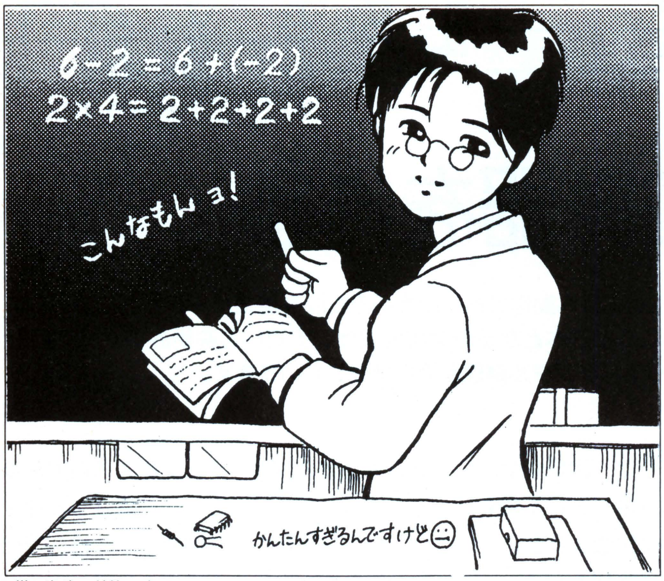


表1 10進↔2進対応表

10進数	2進数
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111



illustlation Y.Kawahara

と考え直してやれば，6に−2を加える加算と見なすことができるのです。ですから，加算器だけで加算も減算もできることになります。ただし，負の数を表現できるように元のデータ形式を工夫しておかなければなりません。

次に乗算について考えてみましょう。これも掛ける数分だけ掛けられる数を足していけば，答えを得ることができます。たとえば，

　　2×4＝8

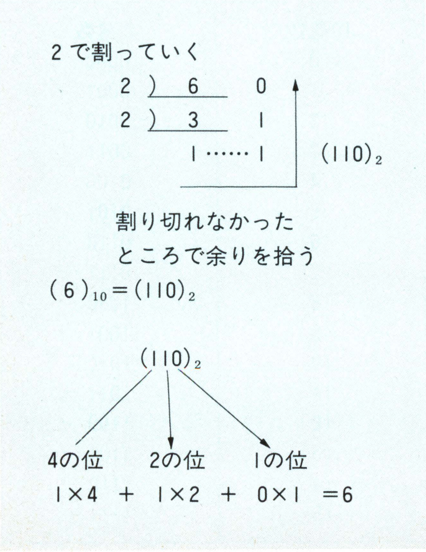
について，2を4個分足すことに置き換えると，

　　2＋2＋2＋2＝8

と加算の形に書き直せるのです。実際には乗除算はALUだけではなく，シフトレジスタという演算器を使って簡単化して行うことが多いのです。しかしながらシフトレジスタで簡単にできるのは，2倍の繰り返しなので2倍，4倍，8倍……の系列以外では，加算を組み合わせて実行することになります。

このように，四則演算といっても加算がすべての基本になっているのです。そこで，これから加算を実現する演算器に絞って話を進めていくことにします。

図2　10進数から2進数への変換



2進数のデータ表現

演算器の理論を説明する前に，コンピュータで数を扱うときのデータ表現形式について確認しておきましょう。といっても難しい話は抜きにして，ここでは皆さんご存じの2進数を復習するだけにとどめておきます。デジタルコンピュータでは，このあとで説明するように電子回路で数を表していくため，基本的には0と1しか扱いません。ということは，0と1のみですべての数を表せる2進数が，データ表現の基本になるのです。この連載では当分の間整数しか使わないので，特に問題はないでしょう。参考まで，表1に10進数で0～15に対応する2進数を挙げておきます。

さて，2進数で10進数を表すときの一般的な変換方法を説明しておきます（図2）。まず，10進数の6を2進数に直したいときは，問題の6を2でどんどん割っていきます。最初に割ると，

　　6÷2＝3 余り0

次にこの答えの3をまた2で割ると，

　　3÷2＝1 余り1

1はもう2で割れないので，ここで止まります。こうして順番に割ってきた余りを後ろのほうから順番に並べた結果$(110)_2$が答えになります。

今度はこの$(110)_2$がどのような組み立てで2進数になっているかを確認します。2進数は下から1の位，2の位，4の位，8の位というように2のベキ乗で並んでいます。したがって，各位の数を足していけば10進数に直すことができます。

　　1×4＋1×2＋0×1＝6

この程度のことなら，皆さんも十分ご承知のことでしょう。しかしながら，この2進数を使った足し算は，今月の中心テーマですのでいまからみっちり考えていくことにします。そして，2進数のデータ表現については負の数をどう扱うかどうかで，また別の問題となります。というのも，先ほど述べたとおり減算は負の数の加算として考えるからです。2進数の負の数は通常「2の補数」という概念を使いますが，今月は加算器に焦点を絞るため負の数の扱いについては，次のステップまで残しておくことにします。

では，正の2進数をデジタル回路で表していくところから押さえておきましょう。

基本論理演算

デジタル論理回路の基本となっているのは，19世紀にイギリスの数学者ブールが考案したブール代数というものです。数学としての厳密な議論は省略しますが，この代数系で出てくる数は0と1の2種類しかなく，またこの論理回路を構成するための論理演算として，AND（論理積），OR（論理和），NOT（論理否定）の3種類で十分なことがわかっています。

論理演算というのは，上で述べてきた算数における算術演算とは別のものだと考えてください。ここで，仮にANDを‘∧’，ORを‘∨’，NOTを‘￣’と表すことにします。すると，このブール代数で考えうる論理演算のANDについては，

　　0∧0＝0
　　0∧1＝0
　　1∧0＝0
　　1∧1＝1

となり，いわゆる普通の算術演算における掛け算と同じです。出てくる数は0と1だけなので，ひとつでも0が出てくると答えは0になり，両方1のときだけ答えが1になります。

そしてORについては，以下のとおりになります。

　　0∨0＝0
　　0∨1＝1

1∨0＝1

1∨1＝1

基本的には足し算と同じですが，両方1のときには答えは2にならず，1になるところが普通の足し算とは違っています。ここが論理演算と算術演算の異なる点です。

NOTについては，

$\overline{0}$＝1

$\overline{1}$＝0

以上のように出てくる数が0か1の2種類しかないので，互いにもう片方に反転させる演算が可能になります。

この3種類の演算を使えば，ブール代数における論理演算のすべてを表現することができます。このあとに出てくるデジタル回路は，どんなものでもこの3種類の組み合わせになっているので，まずこれらの規則だけは確実に頭に入れるようにしてください。

基本デジタル回路

いま述べた基本論理演算を電子回路で実行させるのが，デジタル論理回路です。このとき，論理演算に出てくる0と1を回路に現れる電圧値で表現します。回路全体に印加する電源電圧をHレベル，通常は0VをLレベルとし，論理演算の1/0をH/Lで表現します。

基本的なデジタル論理回路は，TTLやC-MOSといったロジックICを使って実現できます。上で述べた演算のANDとORとは，入力が2つで出力がひとつのユニットになっていて，2つの入力のそれぞれにHかLの電圧をかけたとき，出力に現れる電圧HまたはLがその演算結果となっているのです。

また，NOTは入力，出力ともにひとつずつのユニットになっています。この場合も，入力にかけた電圧H/Lの反転した電圧が出力端子に現れます。

くどいようですが，デジタル論理回路におけるAND回路は，

L∧L＝L

L∧H＝L

H∧L＝L

H∧H＝H

OR回路については，

L∨L＝L

L∨H＝H

H∨L＝H

H∨H＝H

NOTについては，

$\overline{L}$＝H

$\overline{H}$＝L

ということになります。これらはHを1，Lを0と置き換えると論理演算と同じものですから，今後は論理演算の1/0と論理回路のH/Lとを同じものとして説明していきます。これら3種類のユニット回路のことをゲート（門）と呼び，これらのゲートを回路図で書き表すときには，図3のような記号を使います。

さて，それぞれのロジックICの中身は，トランジスタの集まりからできています。ここではなぜトランジスタ回路の組み合わせで論理回路が構成できるか，という仕組みについての説明を省略し，最初からAND, OR，NOTの基本論理ゲートがあるものとして話を進めていきます。ただし，実際のICの活用例については，来月の実習編で詳しく述べることにし，今月は図3の記号を使った概念的な回路図を考えていくことにしたいと思います。

加算器の設計

いよいよ実際に加算器を設計するところにきたのですが，その前に問題となる2進数の加算を復習しましょう。1桁の2進数の加算は繰り上がりを考慮すると答えが2桁になり，

0＋0＝00

1＋0＝01

0＋1＝01

1＋1＝10

となります。1桁の加算であれば，別に10進数に直してみるまでもなく簡単に理解できるでしょう。

ここで繰り上がりの2桁目について着目すると，

入力x, y		出力z
0	0	0
0	1	0
1	0	0
1	1	1

となっており，これはちょうど基本論理演算のANDと一致していることがわかります。論理演算式で書き直すと，

0∧0＝0

0∧1＝0

1∧0＝0

1∧1＝1

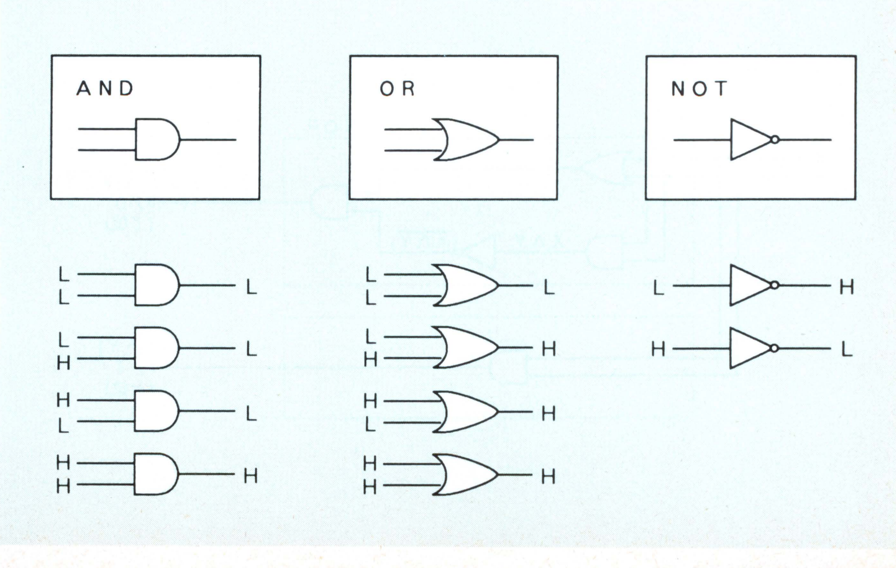
ですから，繰り上がりの部分はロジック回路のAND回路で作ればOKです。

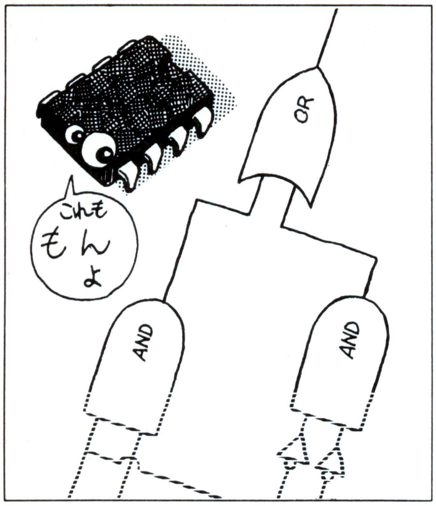
問題なのは1桁の部分です。これも入出力の対応関係で書いてみると，

入力x, y		出力z
0	0	0
0	1	1
1	0	1
1	1	0

となり，一見基本論理演算のORに似ています。しかし，入力が両方とも1のときに0となっているのです。ここで別の見方をす

図3　論理ゲートの記号





ると2つの入力が同じ（どちらも0かどちらも1）ときに0で，違う（0と1の組み合わせ）ときに1という規則になっていることがわかります。そしてこの規則は，XOR（排他的論理和：Exclusive OR）と呼ばれるものに相当するのです。

このため加算器を作るには，このXORを実行する回路を設計しなければなりません。先ほど論理演算においては，すべての論理演算をAND，OR，NOTの基本演算の組み合わせで表現することができるといいました。それにしたがってXORをAND，OR，NOTの組み合わせで表すことを考えていきます。

XORを作る

では，実際にXORを設計してみましょう。ここでは，OR回路をベースに組み立てていくことにします。

$0 \vee 0 = 0$

$0 \vee 1 = 1$

$1 \vee 0 = 1$

$1 \vee 1 = 1$

まず，2つの入力x，yのORを$u = x \vee y$とおいておきましょう。次にこの論理表をXOR（記号'⊕'）の論理表と比較してじっくりにらむと，

$0 \oplus 0 = 0$

$0 \oplus 1 = 1$

$1 \oplus 0 = 1$

$1 \oplus 1 = 0$

入力の両方が1のときだけ出力が反転していることになります。この，「入力の両方が1のときだけ」という部分に着目すると，ANDの論理と同じということがいえるのです。ここでANDの論理をもう一度書きます。

$0 \wedge 0 = 0$

$0 \wedge 1 = 0$

$1 \wedge 0 = 0$

$1 \wedge 1 = 1$

今度は2つの入力x，yのANDを$v = x \wedge y$とおいておきます。ただし，このままでは，XORのように両方が1のときだけ出力が反転することにはつながりません。

ここでvの否定を取り，

$\overline{(0 \wedge 0)} = 1$

$\overline{(0 \wedge 1)} = 1$

$\overline{(1 \wedge 0)} = 1$

$\overline{(1 \wedge 1)} = 0$

uとのANDを取った結果をzとすると，

x	y	u	v	z
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0

目的のXORになり，論理式を書き下すと，

$z = x \oplus y$

$= (u \wedge \overline{v}) = ((x \vee y) \wedge (\overline{x \wedge y}))$

ということになるのです。

実際の加算器の設計

では，実際にTTLICを使って加算器を設計してみることにします。回路図は図4のとおりで，上の位についてはANDゲート1個のみ，下の位については4つのゲート回路（AND 2個，OR 1個，NOT 1個）の組み合わせで出来上がります。下の位についてはXORの論理式，

$z = (u \wedge \overline{v}) = ((x \vee y) \wedge (\overline{x \wedge y}))$

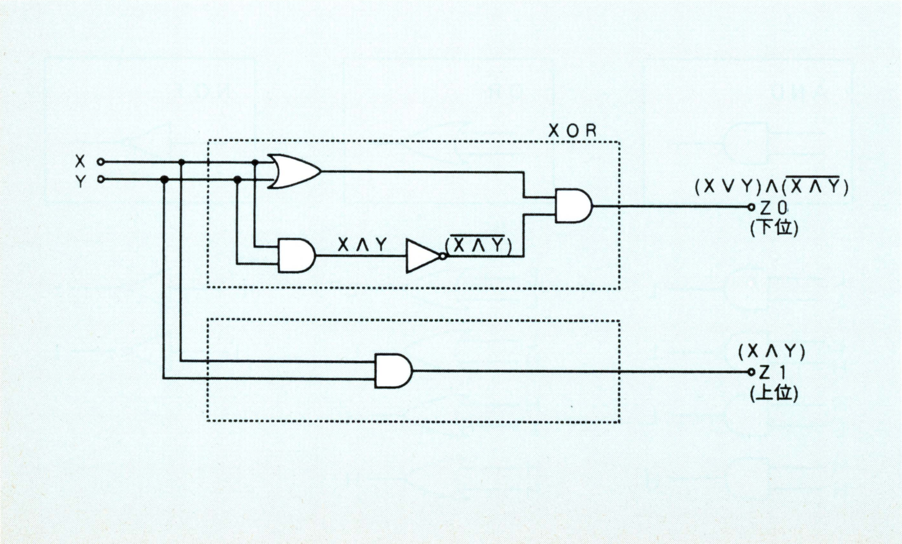
AND 2個，OR 1個，NOT 1個にそのまま対応しているのがわかるでしょう。

この回路図でわかるようにデジタル回路では，ひとつのゲートの出力をほかのゲートの入力に直接つなげることができます。さらに，ひとつの出力から2つ以上の別のゲートの入力へ並列につなぐこともできるのです。しかし，その上限の数は，ICの種類（TTLかC-MOS）によって決まっていますので，そのつど規格表で確認しなければなりません。このあたりの実際的な回路工作上の問題は，来月の実習編で扱います。

回路図にはゲートの入出力間を結んでいる配線に，信号が示す論理式を書き込んであります。最初の入力から信号を追っていって，XORの論理回路をAND，OR，NOTの組み合わせで考えた道筋と，同じようになっていることを確認してみてください。

以上で繰り上がりつき1桁の加算器の設計が出来上がりました。実をいうとこの回路は，いまのままではあまりスマートとはいえません。もちろん間違いではないのですが，実際にTTLICを使って回路を組むときには無駄が多いのです。そこで，この回路の問題点とそれに応じた実際のTTLICの選び方は次回改めて説明しようと思います。そして基板に部品を配置し，配線していく実習を通じて加算器を完成させていきます。では来月またお会いしましょう。

図4 加算器回路図



MUSIC

ZMUSICシステムver.1.10

バージョンアップされたZ-MUSICシステムの内容と追加機能について解説します。今回はMMLレベルでの使い方を中心に，来月はプログラムからの呼び出しを中心に進めていきます。あわせてver.1.0用マニュアルの補足も行いますので参考にしてください。

Nishikawa Zenji 西川 善司

ZMUSICシステムは数々の試練を乗り越えてver.1.10にたどり着き，6月号の付録ディスクに収録することができました。現在MOOK「ZMUSICシステム」の改訂版を執筆中ですが，その前に初版ver.1.0を買ってくれた読者の皆さんのためにver.1.10はver.1.0からどう変わったのか，新設の機能などを解説します。改訂版のほうはもう少しお待ちください。

新しい機能

今回のバージョンアップの内容はデバッグと高速化，そして機能拡張です。これにともないバージョンコードも$11に変更されています。よって，ver.1.1でコンパイルされたデータは従来のZMUSIC.Xでは演奏できません。

バージョンの変化はわずかでも内部は大幅に変わっています。拡張された仕様と主な特徴を挙げてみましょう。

●音量モジュレーションに対応

全体的に高速化された分の余力で音量モジュレーションに対応しました。波形は三角波のみです。

●PCM8に対応

PCM8.Xを組み込んで起動することでAD PCMを8トラックまで扱うことができます。音量は可変でフェードアウトなどにも対応しています。

●相対テンポコマンドの追加

テンポ指定に相対値が使えるようにしました。アチェランドやリタルダンドなどが簡単に記述できます。

●ゲームでの使用時の問題を考慮

もともとゲームでも使えるドライバを目指して開発されていましたが，細かい部分での不都合を解決しました。これはSION IIを見ていただければわかると思います。

●ZMSでの互換性を保持

ZMUSICでの標準データはZMS形式です。仕様拡張のためZMDレベルでは一部のデータでver1.0との互換性がありません。データがおかしいようならver.1.1で再コンパイルしてください。

そのほか，タイマの種類によらずちゃんとしたテンポで曲を演奏できるようになりました。ですからたとえば，タイマA用の曲としてコンパイルしたZMDデータをタイマBでも正常なテンポで演奏することができます。

また，MFP多重割り込み対応モードやPCM8モード時には都合上[COPY]キーがきかなくなります。

PCM8.Xモードについて

PCM8.Xは江藤啓氏により作成されたX68000本体の改造をせずにAD PCM音を8和音までリアルタイムに合成できるアプリケーションです。これも6月号の付録ディスクに収録されています。このPCM8.Xを事前に組み込んでからZMUSIC.Xを組み込むとZMUSIC.Xは自動的にPCM8.Xモードで動作します。ZMUSIC.Xを組み込んだあとにPCM8.Xを組み込むことはできません。

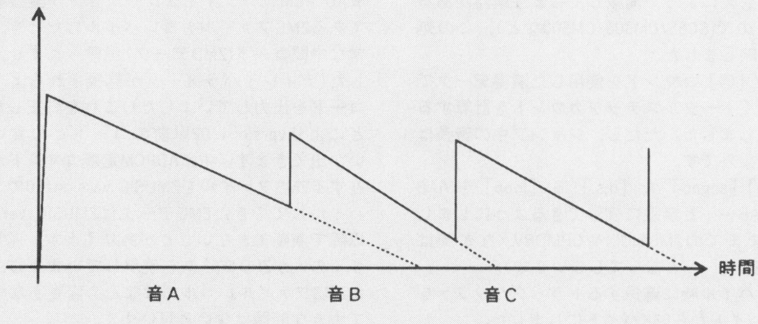
さて，ZMUSIC.XのPCM8モードには2通りあります。

ひとつはAD PCM音源を8つのほぼ独立したチャンネルとして使用するモードです。このモードにするにはZMUSIC.Xを組み込むときに特別なスイッチなどを設定する必要はありません。このモードを特にPCM8独立チャンネルモードと呼ぶことにします。ZMUSIC ver1.10では8本のトラックをAD PCMチャンネルにアサインしてポリフォニックAD PCM演奏が可能です。また，AD PCMに音量指定が可能になります。

もうひとつのモードはPCM8.Xを組み込んだあと'-O'スイッチをつけてZMUSIC.Xを組み込むと設定されるモードです。このモードはPCM8.X用ではない曲，つまりAD PCM 1声用の曲データのAD PCM音を無理やりポリフォニックに演奏するモードです（図1）。

たとえばシンバルを叩いたあとこのシンバルが鳴り終わらないうちにスネアを叩いたとすると，AD PCM 1声の従来の曲ではシンバルの音がブツリと切れてスネアの音に切り替わっていました。ここをブツ切りにしないで（PCM8.Xを用いて）ちゃんと前後の音を重ねて演奏してしまおうというのがこのモードです。このモードはPCM8ポリモードと呼ぶことにします。

図1 PCM8モードの模式図



PCM 8 モードでは点線部の音も発音する

PCM8独立チャンネルモード時の具体的な操作方法に関しては来月号で詳しく解説します。

新設ZMSコマンド

(Ｅch1,ch2,ch3……)

演奏中のチャンネルをリアルタイムにマスク/解除することができます。任意の個数のチャンネル番号を書くことによって指定以外のチャンネルをマスクすることができます。これはZP.Rの'-E'スイッチとまったく同機能のものです。(Bn),m_ch()によってチャンネル番号と対象デバイスは変動する点に注意してください。(Dn)などと同じくデバッグ援助的な機能として設けてあるため，コンパイル時には無視されます(ZMDデータとしては生成されない)。

使用例

(E1,2)

チャンネル1,2以外をマスク

必ず(M tr, size), (A ch, tr), m_alloc, m_assign以降に書いてください。トラック確保，チャンネルアサイン前に記述しても意味を持ちません。(p)などの直前に書くのがスタンダードな使い方でしょうか。

ZP -DとしてZP.Rを常駐させたあとに使用できる再演奏機能([SHIFT]+[XF4])を実行したときにも影響を与えるので，ひとつのチャンネルの演奏チェックを何度も行うときにはとても便利です。

(Ｆｎ)　　-85≦n≦85

正数でフェードアウト。負数でフェードインを行うことができます。この指定を行った時点から演奏中の全トラックがフェードイン/アウトします。ただし，1トラックでもすでにフェードイン/アウトしていた場合は無視されます。

.exclusive

.roland_exclusive

'.exclusive'命令や'.roland_exclusive'命令で ' または " で文字列をくくるとそのASCIIコードを値とすることができるようになりました。ただし漢字の場合はエラーとなり，また終端記号を書かずに改行した場合は文字列の終わりと判断します。

使用例

.exclusive $16,$10= {"APPLE",""}

図2　PMパラメータ正数

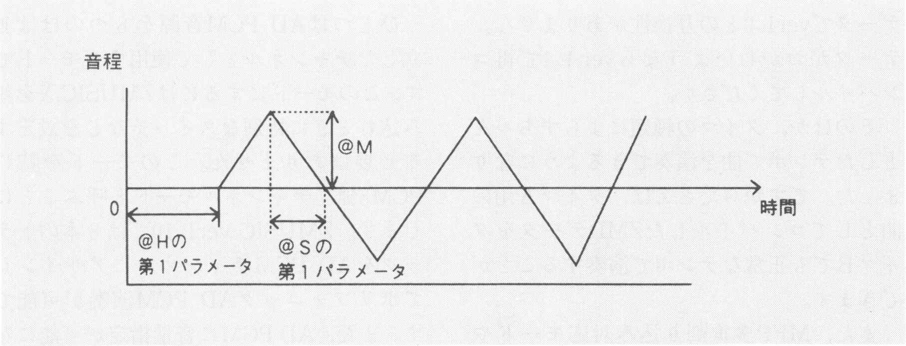
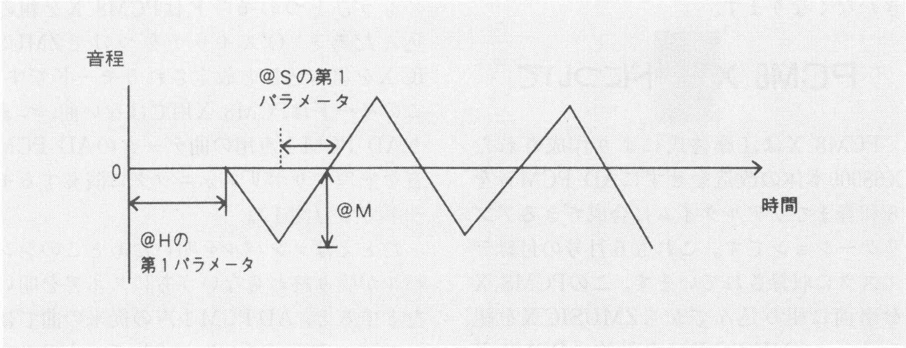


図3　PMパラメータ負数



Z-MUSICの主要変更点

★@m0 @a0 @z0など0というパラメータ1個だけのときは各機能のスイッチオフと解釈するようにしました。

★OPMDRV.Xにあった機能停止/再開スイッチはZMUSIC.Xでは無意味なので削除しました。

★version 1.04以前まではm_play( )時にデフォルト音色として音色番号1をセットしていましたが，ユーザーが設定した音色パラメータが音色切り替えによって消滅してしまう楽器があるようなので(SC55/CM300/CM500など)，この処理を省略しました。

★ [!]/ [@] コマンドを使用した演奏データでも正しくトータルステップカウントを計算するようにしました。ただし，ジャンプ中の音長は計算対象外です。

★ [do] [segno] A [d.s.] B [loop] を AAB AAB AAB……と無限に演奏できるようにしました(いままでのZMUSIC.XやOPMDRV.Xなどでは AAB AB AB……となってしまいます)。

★コンパイル時に確保するトラックバッファを128Kバイトから64Kバイトにしました。

★ZP.Rで拡張子を省略した場合'.ZMD'を自動添付しましたが，これでファイルが見つからないときはさらに'.ZMS'を添付してもう一度探すようにしました。

★ZMUSIC.X -C で拡張子を省略した場合'.ZMS'を自動添付しましたが，これでファイルが見つからないときはさらに'.OPM'を添付してもう一度探すようにしました。

★AD PCMコンフィギュレーションが直接書いてあるZMSファイルをコンパイルしたとき，正常な中間コード(ZMDデータ)が得られませんでした(ディレイパラメータが無視されるようなコードを出力していました)。これを訂正したことによりversion1.02以前のコードとの食い違いが出てきます。中にADPCM定義コマンドが存在するZMSファイルをZMUSIC version1.0でコンパイルしてできたZMDデータはZMUSIC ver.1.1以降で演奏できないことがあります。通常，多くの人がZPDデータを使用していますし，またZMSファイルレベルではなんの変更もないので大きな問題はないと思います。

新設MMLコマンド

以前からあるもので仕様拡張されたものがあります。ここではそれらを含めて紹介します。

●モジュレーション関係

@Mn

ピッチモジュレーション(以下PM)ディレイ≠0のときのディレイモード時の仕様が拡張されました（ディレイ＝0のときの1/8モードの仕様はそのままです)。

＜FM音源の場合＞

パラメータであるピッチモジュレーションデプスの有効範囲が0≦n≦768から-32767≦n≦32768に拡張されました。正数の場合は図2のような波形，そして負数の場合は図3のような波形で音程を震わすようにしました。

＜MIDIの場合＞

変更はありません。有効範囲は前バージョン同様0≦n≦127です。

@An

または,

@An1,n2,n3,n4,n5,n6,n7,n8

＜FM音源の場合＞

@Aで音量モジュレーション(AM，音量LFO，トレモロ)の指定ができるようにな

りました。音量モジュレーション(以下AM)とは単位時間あたりに音量を細かく震わせる演奏表現です。関連コマンドとしてディレイタイム設定コマンドの@H，モジュレーションスピード設定コマンドの@Sがあります。

ピッチモジュレーション@M同様にディレイモードと1/8モードの2つがあります。

AMディレイモード

後述のディレイタイム設定コマンドで与えたディレイカウントのあと，与えた掛かり具合でAMを実行するようになります。与えられるパラメータの個数は1個のみ，範囲は-127≦n≦127です。パラメータが正数の場合は図4，パラメータが負の場合は図5のような波形になります。

AM1/8モード

音長の1/8時間単位で発音後のAMの掛かり具合を変化させることができます。パラメータの範囲は0≦n≦127です。

　　@A60,,,50,127
　　@A,,,127
　　@A,120,-5,＋10

のように省略，相対表現も可能です。パラメータは8個までで省略した箇所は以前のものを持続することになります。こちらのモードはディレイ＝0(@Hの第2パラメータ＝0)のときのみ有効となります。パラメータと波形の概要との関係は図4と同じようになります(ただし必然的にディレイ＝0です)。

＜MIDIの場合＞

ディレイモード，1/8モードともに変更はありません。従来どおり@AはARCCのコントロールに使用できます。

@Hn1,n2

前からこのコマンドは存在しますが一応説明しておきます。n1はピッチモジュレーションのディレイの設定，n2はAMまたはARCCのディレイの設定を行います。どちらか一方の省略ができ，省略されたほうには設定をしません。範囲は0≦n1,n2≦32767です。

使用例

　　@H72,64 両方設定
　　@H,64　AM/ARCCのみ設定
　　@H72　PMのみ設定

@Sn1,n2

第2パラメータでFM音源部分のAM機能のモジュレーションスピードが設定できるようになりました。設定範囲も拡張されn1,n2ともに0≦n1,n2≦16383です。どちらか一方の省略ができ，省略されたほうには設定をしません。

使用例

　　@S8,7両方設定
　　@S,7　AMのみ設定
　　@S8　PMのみ設定

●デバッグ援助機能

［@］

次の［@］まで音階以外のコマンドを高速実行します。次の［@］を見つけた時点で通常演奏に戻ります。その時点まで通常に演奏した状態と同じになっていますので長い曲を作ったりするときには便利です。

前バージョンからある［！］と違うのはあるひとつのトラックで指定するとほかのトラックも同時に高速演奏開始する点です。たとえば，

　　(t1) L8［@］abc［@］def
　　(t2) L8gabcde
　　(p)

とするとトラック1はd，トラック2はcから演奏が開始されます。また，［@］～［@］間に多くの音符やコマンドを含む場合少し待たされることがあります。

●音長関係

＊音長

　　'ceg',＊248,＊12
　　'c＊384eg',＊6
　　(c<e)＊248,＊6
　　(c＊456,b),＊12

といった"＊"による音長表記の自由度を拡張しました。'L＊音長n'の表記も可能になりました。ただし0≦n≦254です。

^n

PC-9801式タイがいままで@Wには使用できませんでしたが，この制約を取り払いました。

音長0

音長0の音符の使用を解禁しました。ただし，操作対象デバイスによって機能の方法が少し違うので注意してください(というかMIDIでしか使いものにならない)。音長0の指定は即値指定と，L/@Lコマンドによるものと2通りあります。

即値指定

　　C＊0

L/@L指定

　　@L0 C
　　L＊0 C

＜MIDIの場合＞

使用例

　　C＊0E＊0G＊0　←CEGの和音が鳴りっぱなしになります

単純にそのチャンネルにノートオンを送るだけです。リズムトラックのようなノートオフ不要トラックのシーケンスに役立ちます。たとえばO2C＝バスドラム，O2D＝スネアドラム，O3C＋＝シンバルとします(ちょうどローランド系のリズムキット)。

　　L4O3C＋＊0｜：4 O2 CDCD：｜

とすると最初の小節の頭だけシンバルが鳴り，2回目以降は鳴らないといったようなシーケンスができます。

また1トラックにつき制限なくいくつで

図4　AMパラメータ正数

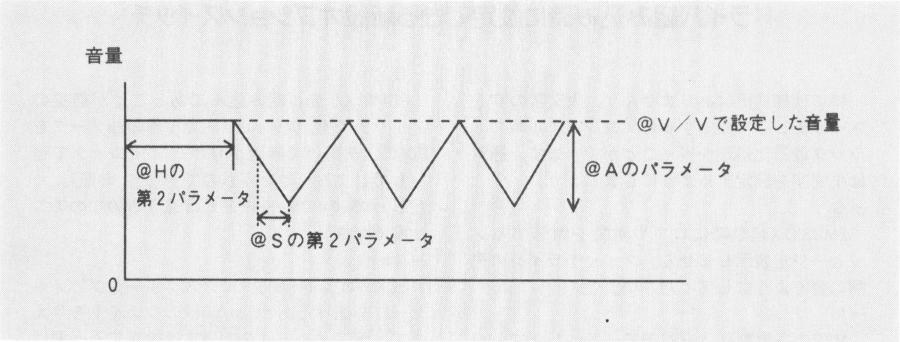
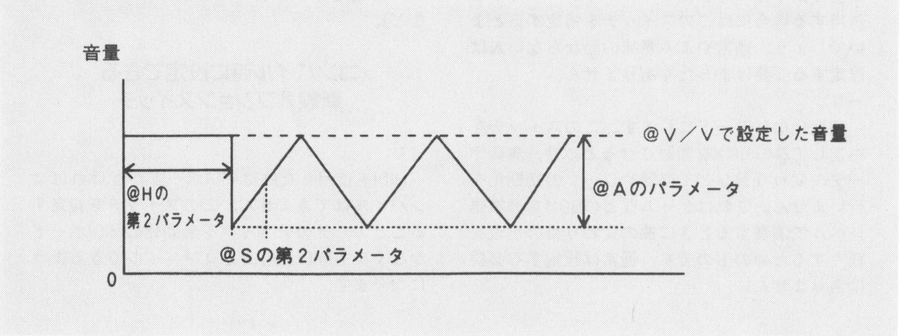


図5　AMパラメータ負数



も記述することが可能です。

音長0の音符に('&'をつけて)タイ指定をして通常の音符へつなげると，その通常音符のノートオフ時にタイでつないだ音符も一緒にノートオフしてくれます。

使用例

C＊0&E＊0&G4　←CEGの和音が4分音符分鳴ります

和音コマンドでは各構成音に対してベロシティのバラツキを設定できませんでしたが，

@U110C＊0&@U105E＊0&@U88G4

のように記述することによって，それが可能となりました。

この書式では最大8和音まで記述できます。最大8音のときは音長ゼロの音符が7つと通常音長の音符がひとつという構成になります。音長ゼロの音符を8つ以上記述した場合はノートオフされない音が出てきます。また，従来の和音コマンドとの混在はもちろん可能です。ですから状況に応じて使い分けができます。また，この2つの書式間のタイ/スラーも表記可能です。

<内蔵音源の場合>

基本的にモノフォニックです。この表記でMIDIのように和音を表記することはできません。和音を実現したいときは専用コマンドを用いてください。音長ゼロがFM/AD PCMではどういった効果をもたらすかを簡単に解説しますと，

C＊0E＊0G＊0R1

とするとGの音が全音符音長で演奏されます。前の2つのC，Eは無視されます。タイ/スラー指定をしても同様です。ですから，

C＊0&E＊0&G4

とした場合はG4のみ発音されます。

●ベロシティ関係

U

リアルタイムに考慮される(ループコマンドなどの影響を受ける)相対ベロシティとしてMML "U" を新設しました。値の範囲は-127≦n≦＋127です。

使用例

@u127 ｜：3 c u-10 ：｜ d

この例では@u127，@u117，@u107でcが演奏され@u97でdが演奏されます。

@u5 ｜：3 c u+5 ：｜ c

この例では@u5，@u10，@u15でcが演奏され@u20でdが演奏されます。

±を付けなかった場合は@uとまったく同じに機能します。たとえばU127と@U127はまったく同じに機能します。

なお，管理はまったく"@U"とは別です。ですから，

@u5 ｜：3 @u+50 c u-5：｜ d

としたとき@u55で3回cが演奏され@u50でdが演奏されます。U,@Uを使い分ければ，かなり複雑なシーケンスができると思います。増減によってパラメータが許容範囲を超えてしまうような場合，最大値，最小値に自動修正されます。

また，パラメータを省略したとき，すなわちUのみ記述したときは以前設定した@uを再設定することができます。相対ベロシティで変化してしまったベロシティをリセットするのに便利です。

使用例

@u120 ｜：10 c u-10 ：｜ d u e
↑
@u120と同等の機能を果たす

この例ではcが@u120，@u110，@u100……，dが@u20，eが@u120で演奏されます。

ベロシティシーケンス実行中にこのコマンドを設定するとベロシティシーケンスは強制的にスイッチオフとなります。

●ベロシティシーケンス

シーケンス最大値が8から16に拡張されました。

●テンポ関係

相対テンポコマンドが新設されました。リアルタイムに考慮されるためループ内の使用でも意味を持ちます。

T±n

テンポを±n増減します。nの範囲は-32768≦n≦32767です。

@T±n

テンポを割り込みタイマ値レベルで±n増減します。タイマのことについて熟知している人以外は使用しないほうが無難です。nの範囲は-32768≦n≦32767です。

(注意) 増減によってテンポの許容範囲を超えてしまった場合は最大値，最小値に修正されます。

使用例

｜：10 T+10 CDE：｜　…だんだんテンポが速くなる

｜：10 T-10 CDE：｜　…だんだんテンポが遅くなる

●その他

I n1,n2

音色メモリを複数版持ったMIDI楽器へどのバンクをセレクトするかを設定するものです。前バージョンではn2を省略できませんでしたが本バージョンより省略できます。たとえば，

I10

としたときには，

I10,0

として機能します。

? n1,n2

ZMUSICの演奏ワークを直接書き換えるコマンドです。第1ワークに書き換えるワークオフセット値，第2パラメータにデータを与えます。ワークオフセットについてはMOOKの増刷改訂版のディスクについてくる"LABEL.S"を参照してください。

ドライバ組み込み時に設定できる新設オプションスイッチ

－C

特に仕様変更はありませんが，大文字の'C'をスイッチとして指定すると，コンパイル時のトラック最適化処理を省くことができます。通常は小文字を設定するようにしましょう。

－G

ZMUSIC.X起動時にロゴや常駐を報告するメッセージを表示しません。スイッチラインの先頭に書くようにしてください。

－M

MFPの多重割り込み対応モードにします。ラスタースクロールなどの処理をジャマしません。高度な割り込みを駆使したアプリケーションと併用する場合にはこのスイッチを指定するとよいでしょう。通常やよく意味のわからない人は設定する必要はまったくありません。

－N

初期化なしモードにします。このスイッチを指定してZMUSIC.Xを常駐させると以後，演奏データの変わり目などで楽器やワークの初期化を行いません。これはゲームなどのBGMを機械語レベルで演奏するときに曲の変わり目の処理を軽くするためのものです。通常は設定する必要はありません。

－O

PCM8.Xが先に組み込んであることが前提のスイッチです。従来のAD PCM 1声の曲データをPCM8.Xを用いて無理やりポリフォニックで鳴らしてしまおうというものです(図1参照)。ただし，本来のPCM8モードとは違うものなのでご注意ください。

－Xn

EOX（エンド・オブ・エクスクルーシブ）メッセージを送ったあと，n/60秒のウェイトを与えます。デフォルトは2で通常は設定する必要はありませんが，極端に応答の遅いMIDI楽器に対しては設定するとよいでしょう(U110/220，D70など)。

コンパイル時に設定できる新設オプションスイッチ

－U

MIDIを使用した曲はMIDIボードがなければコンパイルはできません。このスイッチを指定することで，そのチェックを無効化しMIDIボードがなくてもMIDI使用曲をコンパイルできるようになります。

W

Wn

演奏制御をトラック間で強制的に行うコマンドです。Wのみでほかのトラックより同期信号がくるまで演奏を中断します。

W<トラック番号：1≦tr≦80>でそのトラックへ同期信号を発信し，そのトラックは演奏を再開します。

使用例

(t1)　@1 V15 O4 Q8 CDEF W2 ←トラック2へ同期送信

(t2)　W @1 V15 O4 Q8 GAB<C
↑どこからか同期信号がくるまで演奏を中断

この例では結果的にはCDEFGAB<Cが演奏されます。

(注意)

コンパイルによってトラック番号が最適化されると，うまく同期が取れなくなります。たとえば，

(m1,1000)(a1,1)
(m2,1000)(a2,2)
(m3,1000)(a3,3)
(t1)　@1 v15 o4 q8 cdef w3
(t2)
(t3) w @1 v15 o4 q8 gab<c

の場合，コンパイルの最適化によってトラック2は消去されトラック3がトラック2になってしまうため，トラック1の同期信号は意味をなさなくなります。

また，

(m1,1000)(a1,1)
(m2,1000)(a2,2)
(m3,1000)(a3,3)
(t1)
(t2)　@1 v15 o4 q8 cdef w3
(t3) w @1 v15 o4 q8 gab<c

の場合も同様にトラック1が消去されトラック2が1，トラック3が2へと詰められるので同期コマンドは意味を持たなくなります。ですから，同期コマンドを使用した曲を作成する場合は無意味なトラックを確保してはいけません。

*　　*　　*

今回は新しいコマンドについての解説だけになってしまいましたが，来月は効果音モードやゲームでの使い方などもっとつっこんだ活用法を解説します。お楽しみに。

支援ツールについて

ZPCNV.R

★こんな質問がありました。

1=snare1.pcm　…(1)
1=snare1.pcm　…(2)
.erase　1

としたとき，(2)のほうは削除されますが(1)のほうは削除してくれないみたいです。どうしてですか。

これは(2)のオーバーライトされた時点で(1)の情報はすべて失われてしまうので最後のerase命令は(2)に対してしか機能しないわけです。基本的にオーバーライトは反則なわけです。ですから本バージョンからオーバーライトのメッセージを反転色で表示することにし，注意を促すようにしました。

★$feff(65279)バイトを超えるデータを許可するようにしました(PCM8.Xモード時に使用可)。

★最近拡張子'CNF'がグローバルなものになってきたようなので，拡張子省略時にはこれを自動添付するようにしました。

ZP.R

★ZP.Rのジュークボックス機能の仕様を変更しました。必要ファイルを先読みし，曲の変わり目にディスクを読まないようにしました。仕様変更の理由は「割り込みでディスクを読むのは気持ち悪い」という意見が多かったためです。仕様は以下のようになり，ZP.X ver.1.1では割り込みでディスクを読むようなアクロバットがなくなりました。

1) ジュークボックス用INDEX FILEにはZMDデータのみが有効(拡張子は省略可)。ZMD以外を記述した場合はエラー。表記方法に変更なし(繰り返し回数(省略可)，filenameの順)。

2) ジュークボックスで演奏させる曲データにはZPDデータ指定('.adpcm_block_data')以外のAD PCMデータ登録文が存在してはならない。存在した場合や'.adpcm_block_data'が複数あった場合はエラーとなる(1個のみ有効)。

3) ジュークボックスで演奏させる曲データにMDDファイル転送命令('.midi_dump')が複数あった場合はエラーとなる(1個のみ有効)。

4) ジュークボックスで演奏させる曲データ中にAD PCM定義コマンド('.Onk=filename,Pn,Vn,Mn,Dn')やAD PCMコンフィギュレーションファイル実行コマンド('.adpcm_list')を使用していた場合はエラーとなる。

その他，注意点としては，

・先読みするのはZPDデータとMDDデータです。巨大サイズのZPDデータなどをたくさん読み込んだ場合，メモリが不足することもあるので注意してください。

・DMAアクセスやディスクアクセスなどの邪魔をしないので音楽を聴きながら別の作業が安心して行えます。

★ZP.Rのジュークボックスに強制的に次の曲へ移る機能をつけました。

[SHIFT]＋[OPT.1]　　すぐ次の曲へ
[SHIFT]＋[OPT.2]　　F.O.後次の曲へ

キースキャンは処理の都合上，約1秒に1回ほどしか行っていませんのでご了承ください。

★ZP -Fn でマイナス値の使用を許可(フェードインの指定が可能)。

★チャンネルマスク機能である'-E','-M'がPCM8.X対応になりました。PCM8.X組み込み時はm_ch( ),(Bn)によらずAD PCMの2〜8チャンネルが便宜上チャンネル26〜32に対応します。

〈使用例〉

(b0)のときADPCM1と4チャンネルをマスクしたい場合。

A>ZP -m9,28

(b1)のときADPCM1と2チャンネルをマスクしたい場合。

A>ZP -m25,26

★新規スイッチとして'-O'がつきました。任意のチャンネル音量を通常より小さくすることができます。'-O'の後ろの第1パラメータが出力割合(0〜128)，第2パラメータ以降が操作対象チャンネルです。出力割合128で通常に戻ります。また'-O'の後ろになにも数値指定をしなければ前チャンネルが通常出力に戻ります。

なお，操作対象チャンネルはm_ch()や(Bn)で変動します。PCM8.X組み込み時はm_ch(),(Bn)によらずAD PCMの2〜8チャンネルが便宜上チャンネル26〜32に対応します。FM音源の場合は変化が指数関数的であるので注意してください(第1パラメータ＝90付近でほとんど消音してしまいます)。

〈使用例〉

(b0)のときFM音源チャンネル1,3,6の出力割合を80にしたいとき

A>ZP -o80,1,3,6

(b0)のときMIDI1チャンネルとADPCM1と2チャンネルの出力割合を99にしたいとき

A>ZP -o99,10,9,26

★ジュークボックス使用時にはZMUSIC.X側の各バッファはいくらでもよいことにしました。

A>ZMUSIC -t1 -w1
(AD PCM BUFFER＝0KB，トラックバッファ＝1KB，ワークエリア＝1KB)

で組み込んでもジュークボックスのほうで必要なデータを読み込んで常駐してしまうため，ドライバ側には各データのポインタのみを教えてやるだけでいいわけです。

★ZPDファイルの登録が行えなくなっていたのを修正しました。

★デバッグモード時のm_play()は新設のファンクション$0f 'm_play2'で行うようにしました。

m_play(1,2)

としたあとに[SHIFT]＋[XF4]とすると，いままでだと全トラック演奏してしまいましたが，本バージョンから，

m_play(1,2)

を再度実行してくれます。

MON.R

★MON.R実行中スペースキーを押すと，ポーズをかけられるようになりました。

★チャンネルのマスク機能の対応デバイスが，(Bn)またはm_ch( )コマンドによって変動するのが不評だったので，絶対的に対応するように仕様を改良しました。

テンキー	[1]〜[9]	FM1〜8,ADPCM
ファンクションキー	[F1]〜[F10]	MIDI1〜10
フルキー	[1]〜[6]	MIDI11〜16
フルキー	[Q]〜[I]	ADPCM1〜8
	[DEL]	マスク解除
	[HOME]	マスク反転
	[SPC]	ポーズ

上記のようなキー操作で簡単に出力チャンネルのコントロールができます。

ZMUSICシステム　マニュアルの訂正

★P.13,P.30 フェードアウトコマンドのパラメータの範囲は-85≦n≦85です。

★P15 u220_timbre命令の説明が間違っています。以下に正しいものを示します。

関数名	u220_timbre(tm,name,ary,id)
機能	U220のティンバーパラメータの設定
引数	tm＝ティンバー番号(char:1〜128) name＝ティンバーの名前(str:12文字以内) ary＝パラメータ(dim char ary(25)＝ {pr1, …, pr26}) id＝u220のデバイスＩＤ(char)
戻り値	なし
備考	配列は必ず１次元，char型で添え字は必ず25でなければならない。 IDは省略可能。省略すると以前設定したものが選択される。最初の使用時に省略した場合はドライバ内のデフォルト値が選択される。

ary(0)：Tone Media	(0〜31：1,1〜31)
ary(1)：Tone Number	(1〜128)
ary(2)：Timbre Level	(0〜127)
ary(3)：Velocity Sens	(1〜15：-7〜+7)
ary(4)：Channel Press Sens	(1〜15：-7〜+7)
ary(5)：Env Attack Rate	(1〜15：-7〜+7)
ary(6)：Env Decay Rate	(1〜15：-7〜+7)
ary(7)：Env Sustain Level	(1〜15：-7〜+7)
ary(8)：Env Release Rate	(1〜15：-7〜+7)
ary(9)：Pitch Shift Coarse	(8〜56：-24〜+24)
ary(10)：Pitch Shift Fine	(14〜114：-50〜+50)
ary(11)：Bend Range Lower	(0〜15：-36,-24,-12〜0)
ary(12)：Bend Range Upper	(0〜12)
ary(13)：Channel After Sens	(0〜27：-36,-24,-12〜+12)
ary(14)：Poly After Sens	(0〜27：-36,-24,-12〜+12)
ary(15)：Auto Bend Depth	(0〜27：-36,-24,-12〜+12)
ary(16)：Auto Bend Rate	(0〜15)
ary(17)：Detune Depth	(0〜15)
ary(18)：Rate	(0〜63)
ary(19)：Waveform	(0〜8)
ary(20)：Depth	(0〜15)
ary(21)：Delay	(0〜15)
ary(22)：Rise Time	(0〜15)
ary(23)：Modulation Depth	(0〜15)
ary(24)：Ch After Sens	(0〜15)
ary(25)：Poly After Sens	(0〜15)

(U220のマニュアルP.65,P.149参照)

★P.23の「@Ｔ」コマンドのテンポとタイマ値の相関式が「タイマＡ，Ｂ」に入れ替わっています。正しくは，

タイマＡ＝1024-(78125/テンポ)

タイマＢ＝256-(78125/テンポ)/16

です。

★P.24「@Ｎ」コマンドの説明

文中の「10〜15」は「10〜25」の誤りです。

★P.28右の上から３行目の「その４カウント後にbが発音され，」は「〜eが発音され，」の間違いです。

★P.30「@Y」コマンドの説明

見出しの@Ya1, a2, d3, d4は@Ya1, a2, d1, d2の間違いです。

★P.30「@O」コマンドの解説を訂正します。

@On

OPMのスロット32からノイズを発声する。パラメータの範囲は0≦n≦31で値を省略するとノイズモード解除。なお値とノイズ周波数には，

noise(Hz)＝125000/n

という関係がある。

★P.30「@F」コマンドの説明

文中の「f=」は「n=」の間違いです。

★P.31「[!]」コマンドの動作解説

番号を示す数字が 1, 4, 5, 6 になっていますが 1, 2, 3, 4 の間違いです。

★P.43の「m_tempo」の項の備考のところの「a0. l=現在のタイマを書く」は「a0. l=現在のタイマ値を返す」の間違いです。

★P.44 ファンクション$13, $14の引数の説明が誤っていました。

d3.hw＝pan(0-3)　→　d3.lb＝pan(0-3)

d3.lw＝frq(0-4)　→　d3.hb＝frq(0-4)

ちょうどIOCS _ADPCMOUTのd1. wに相当します。

★P.46〜47の「(DATA FORMATはMEASURE3参照)」は「(〜MEASURE4参照)」の間違いです。

★P.48 ファンクション$3aの説明文中「m_ch("MIDI"),(B0)」は「m_ch("MIDI"),(B1)」の間違いです。

★P.48 ファンクション$3bの機能説明が間違っていました。正しくは今月号P.55を参照してください。

★P.48 ファンクション$43〜44の説明が抜けていました。今月号P.55を参照してください。

★P.49 AD PCMコンフィギュレーションコマンドのZMDコードの説明が誤っています。正しくは下記の表を参照してください。

新設コマンドのZMDコード

マニュアルに誤って記されていたものもここに再掲載してあります。この節は普通に音楽プログラムを入力して楽しむ人にはまったく関係ありませんので読み飛ばしてくださって結構です。

●共通コマンド

AD PCMコンフィギュレーション

. OnK=filename,Pp,Vv,Mm,Dd

$40(.b),note number(.b),pitch parameter(.w),volume parameter(.w),mix note number(.w),mix delay parameter(.w),filename,0(.b)　　合計不定

. Onk=. Onk,Pp,Vv,Mm,Dd

$40(.b),note number(.b),pitch parameter(.w),volume parameter(.w),mix note number(.w),mix delay parameter(.w),0(.w),操作対象ノート番号(. w)　　合計14バイト

テンポコマンド

@T	$90(.b),タイマ値(. w)	合計3バイト
T	$91(. b),テンポ値(. w)	合計3バイト

ワーク直接書き換えコマンド

? offset,data

$d5(.b),offset(.b),data(.b)　　合計３バイト

同期コマンド

シンクロ待ちW

$83(.b)　　合計１バイト

シンクロ送信Wn

$af(.b),トラック番号n(.b)　　合計２バイト

ただしトラック番号nはパラメータとして与えた値－１の0〜79

ＦＭ音源のモジュレーション・スピード・設定コマンド

FM音源部のピッチモジュレーション，アンプリチュードモジュレーションのスピード設定コマンド@Ｓのパラメータの設定範囲の拡張にともないZMDコードも変更されました。従来のも有効です。

$d6,PMのスピード(. w),AMのスピード(. w)　合計５バイト

相対ベロシティ

U+n	$ca(.b),0〜127(.b)
U-n	$cb(.b),0〜127(.b)

相対テンポ

@t+n	$92(.b),n(.w)
@t-n	$93(.b),n(.w)
t+n	$94(.b),n(.w)
t-n	$95(.b),n(.w)

ZMUSIC.X -E(外部シーケンサ同期モード)で組み込んだZMUSIC.X上で，(Zn)コマンドを用いたZMDデータを演奏するとMIDIタイミングクロックが正常な時間間隔で送られていませんでした(ZMSは大丈夫なんですが)。このデバッグにともなってベースカウント設定コマンド"(Zn)"のZMDコードを変更します。

旧　　　$42(.b),tempo base counter(.l)　　　合計５バイト
新　　　$42(.b),master clock(.b),tempo base counter(.l) 合計６バイト
　　　　master clock:(Zn)のnの値そのもの
　　　　tempo base counter:master clockから算出されるタイマ値計算用の定数

音長０の音符

タイをつけない	$ad(.b),note number(.b)	合計２バイト
タイをつける	$cd(.b),note number(.b)	合計２バイト

新設ファンクションコール

初版本のマニュアルで誤っていたものは訂正して再掲載しています。また，仕様変更/拡張がなされたものもあるので注意してください。なおこの節は初心者の方は読み飛ばしてくださって結構です。

ファンクション$0f m_play2
機能　前回に実行されたm_playをもう一度実行する
引数　なし
戻り値　なし
備考　前回のm_play実行時のパラメータでもう一度m_playを呼び出すだけ。よってfunc $11などで演奏を開始したあとこのファンクションを実行しても無意味(実はZP.Rのデバッグモードのためにつけたファンクションなんです)

●func $13,$14の仕様を拡張しました。d3のビット16〜23に効果音の優先レベルを設定できるようにしました。これにより，たとえば先に鳴っている効果音が新しく鳴らそうとしたものより優先度が高い場合，新しく鳴らそうとした効果音のほうを自動キャンセルし，先に鳴っているものを継続します。ただしPCM8.Xモードのときは意味をなしません。
優先レベルの例

現在の優先レベル	新しい音の優先レベル	結果
0	0	新しいのが鳴る
0	1	新しいのが鳴る
1	0	新しいのが拒否される

通常は0(従来どおり)でかまわないでしょう。ゲーム等の面と面のつなぎのメッセージやイベントのメッセージなどの爆発音等の効果音などに邪魔されたくないときに使用するといいでしょう。

ファンクション$13 se_adpcm1
機能　AD PCMの効果音演奏
引数　a1.l＝AD PCMデータアドレス
　　　d2.l＝AD PCMデータサイズ
　　　d3(bit00〜07)＝PAN(0-3)
　　　d3(bit08〜15)＝FRQ(0-4)
　　　d3(bit16〜23)＝優先レベル(低0〜255高)
戻り値　なし

ファンクション$14 se_adpcm2
機能　ZMUSIC内に登録済みのAD PCM音の効果音演奏
引数　d2.l=ノート番号(0〜255)
　　　d3(bit00〜07)＝PAN(0-3)
　　　d3(bit08〜15)＝FRQ(0-4)
　　　d3(bit16〜23)＝優先レベル(低0〜255高)
戻り値　なし

ファンクション$3b set_loop_time
機能　全トラックがd2.b回以上ループしたらa1.lへBSRする
引数　d2.b＝0 loop time(s) (0〜255)
　　　a1.l＝飛び先アドレス(＝0で解除)
戻り値　なし

ファンクション$40　release_support
機能：　ZMUSIC.X用のサブプログラムの名前を登録する
　　　(あるいは登録したものを取り消す)
引数 ：登録時
　　　a1.l≠filename address　(解除プログラムの名前を登録)
戻り値：d0.l＝0≦result code　正常登録された(ok)
　　　d0.l=-1　すでに４つ登録されており，これ以上の登録はできない(error)
備考：　名前を登録しておくとZMUSIC.X解除時に一緒に解除する。filename(100文字以内)は
　　　'a:¥bin¥zp -r',0
　　　のようなフォーマットとする
　　　最大4つまで登録可能
引数：　キャンセル時
　　　a1.l＝0
　　　d2.l＝登録終了時に返ってきたresult code
戻り値：d0.l＝0　キャンセル完了(ok)
　　　d0.l＝-1 エラー
備考：　result codeにより任意のプログラムの登録キャンセルが可能

ファンクション$43　picture_sync
機能　映像同期モードのオン/オフ
引数　d2.l＝0 モードオフ
　　　d2.l≠0 モードオン
備考　使用方法についてはMOOKP.53参照

ファンクション$44　mask_channels
機能　演奏チャンネルのマスキングを行う
引数　d2.l＝マスクしたいチャンネルのビットパターン
　　　bit＝1 mask ch
　　　bit＝0 enable ch
備考　m_ch("fm")または(B0)のときはビット0〜24がFM1〜8,ADPCM1,MIDI1〜16に対応しm_ch("midi")または(B1)のときはビット0〜24がMIDI1〜16,FM1〜8,ADPCM1に対応する。m_ch(),(Bn)によらずPCM8.X組み込み時はビット25〜31がADPCM2〜8に対応する。ＰＣＭ８モード(独立チャンネルモード)に対応した

ファンクション$45　buffer_info
機能：　各バッファのトップアドレス，サイズ，終了アドレスの情報を返す
引数 ：　なし
戻り値：00(a0)＝track buffer top
　　　04(a0)＝track buffer size
　　　08(a0)＝track buffer end(ZMUSIC.Xの最終アドレスに相当)
　　　12(a0)＝AD PCM buffer top
　　　16(a0)＝AD PCM buffer size
　　　20(a0)＝AD PCM buffer end
　　　24(a0)＝work area top
　　　28(a0)＝work area size
　　　32(a0)＝work area end

ファンクション$46　set_zpd_tbl
機能：　ZPDデータの情報を登録する
引数：　a1.l＝ZPD情報テーブルのアドレス
　　　(a1.l＝ZPDファイルの先頭アドレス＋8に相当)
戻り値：なし
備考：　AD PCM bufferの先頭からノート番号ゼロから順番に，
　　　AD PCM DATA ADDRESS(.L),AD PCM DATA SIZE(.L)……
のように格納されている。AD PCM bufferの先頭アドレスはfunc $45で求めることができるので，まったく外部のプログラムからAD PCMデータの差し替えをすることもできる

ファンクション$47　set_output_level
機能　演奏チャンネルの出力割合を設定する
引数　d2.l＝変更したいチャンネルのビットパターン
　　　bit＝1 変更希望チャンネル
　　　bit＝0 変更しないチャンネル
　　　d3.b＝0〜127　　(出力レベル。0は消音)
　　　d3.b＝128以上　(通常に戻す)
備考　m_ch("fm")または(B0)のときはビット0〜24がFM1〜8,AD PCM1,MIDI1〜16に対応し，m_ch("midi")または(B1)のときはビット0〜24がMIDI1〜16,FM1〜8,AD PCM1に対応する。m_ch( ),(Bn)によらずPCM8.X組み込み時はビット25〜31がAD PCM2〜8に対応する。FM音源の場合は変化が指数関数的であるので注意(d3.b＝90付近でほとんど消音する)

ファンクション$48　eox_wait
機能　EOX送信後のウェイト時間を設定する
引数　d2.l＝ウェイト値(0-65535)
備考　d2.l＝0　　　ウェイトなし
　　　d2.l＝1　　　ウェイト最小(約0.017秒)
　　　d2.l＝65535　ウェイト最大(約1092秒)

SIONIIを読む人に贈る

ゲーム内部のイロハ

今回は，SIONⅡで使われたゲームを作るための基礎的な知識を紹介していきます。普段あまり気にしないゲーム内部の事情ですが，自分でゲームを作成しようとして行き詰まってしまったとき，SIONⅡのソースリストを読もうとしたときに，この記事を読み返してみるといいでしょう。

Hamazaki Masaya **浜崎 正哉**

今月は，ゲームを作るための基本的な概念とSIONIIで使われた3Dベクトル移動ルーチンのアルゴリズムを解説していきます。市販ゲームのようなものを作るにしても，ピコピコゲームを作るにしてもゲームの根底にあるものは一緒です。やっぱり基本は大切です，ということで説明することにしました。

なお，前半部分でやたら力説している箇所がちらほらあります。これは，人に見せることを考えてプログラムを作成しようとする人を対象にしていますので，暇プロ的にプログラムを作ろうとしている人は「ま，そんなものかねえ」と軽く流して読み進めていってください。

ゲームの基礎

ひと昔前には，「マルチタスクでもないのに，なぜ一度にたくさんのキャラクターを動かすことができるのか？」といわれたこともありました。いまとなっては当たり前のことですが，要するに内部的に個々のキャラクターを動かしていても，処理速度が高速であれば人間にとっては同時に動いている，と感じさせることができるのです。一般的には，動かすことのできるキャラクターの最大個数分のワークエリアを確保し，キャラクターが出現したときにはワークエリアに必要な情報を書き込んでいきます。そして，個々のキャラクターを管理するルーチンによってそのキャラクターを動かしたり，条件によってさまざまなアクションを起こさせるのです。

ワークエリアの管理

さて，ワークエリアを用意する，といってもただ単にメモリを確保しただけではなんの意味も持ちません。重要なのはそれぞれのワークエリアに意味付けを行うことです。SIONIIでは表1のようにひとつの敵キャラクターに対して，142バイトのワークを確保して管理をしています。

見てのとおり0バイト目(flag)が，そのワークエリアにキャラクターがいるかのフラグ（値が0だとキャラクターがいない）と，敵キャラクターの種類を格納しています。敵の出現ルーチンでは，ここの値が0のワークエリア（キャラクターがセットされていない）を探して敵のデータをセットすることになりますし，メインルーチンでは，flagの値によってそれぞれの敵に対するメインルーチンへ処理を移すことになります。そして，メインルーチンでは，各々のワークエリアを書き換えながら処理を進めているのです。

たとえば，そのキャラクターにアニメーションをさせたい場合，8バイト目の3Dキャラクターデータアドレス(chraddr)と32バイト目のアニメーションフラグ(animef)を用いることになります。

もちろんあらかじめアニメーションフラグ（animef）を用いることになります。
アドレスを格納したテーブルも作成しておく必要があります。そうして，animefの値からテーブルに格納されたデータアドレスを取り出し，chraddrへ取り出したデータアドレスをセットするのです。

あとは，表示プロセス（objectput）で定義するキャラクターのアドレスをMAGICに渡すようにしてやれば，3D物体が画面に表示されます。このプロセスを連続的に行えば，簡単にアニメーションを実行できるのです。

キャラクターの管理

ひとつのキャラクターに対するワークエリアの管理から，もう少し全体を見つめたキャラクター種類ごとの管理を説明しましょう。キャラクターを種類別に管理するためには，図1のようにワークエリアをキャラクターの種類別にするか，キャラクター

表1　基本データ詳細図(142バイト)

オフセット値	サイズ	意味	ラベル
0	W	フラグ	flag
2	W	X座標	x
4	W	Y座標	y
6	W	Z座標	z
8	L	3Dデータアドレス	chr_addr
12	W	X方向の移動量	x_m
14	W	Y方向の移動量	y_m
16	W	Z方向の移動量	z_m
18	W	ベクトル移動用進行カウンタ	vect_cnt
20	W	弾の発射カウンタ	tama_cnt
22	W	弾の発射カウンタ保存	tama_cntw
24	W	X方向の当たり判定サイズ	x_atari
26	W	Y方向の当たり判定サイズ	y_atari
28	W	Z方向の当たり判定サイズ	z_atari
30	W	敵のかたさ	katasa
32	W	アニメーションフラグ	anime_f
34	W	アニメーションカウンタ	anime_c
36	W	ホーミングミサイルカウンタ	hmis_c
38	W	ホーミングカウンタ保存	hmis_w
40	W	動きのフラグ	ugoki_f
42	W	動きのカウンタ	ugoki_c
44	L	ベクトル連続移動用アドレス	vect_addr
ベクトル移動用ワーク×3			
48	W	基準座標フラグ	vect_dat
50	W	差分	vect_dd
52	W	基準の差分	vect_st
54	W	増分（移動量）	vect_id
56	W	商（カウンタ）	vect_shou
58	W	余り	vect_amari
MAGIC用パラメータ			
88	W	X	x_p
94	W	Y	y_p
100	W	Z	z_p
106	W	DX	dx
112	W	DY	dy
118	W	DZ	dz
124	W	HEAD	head
130	W	PITCH	pitch
136	W	BANK	bank

番号を種類別にするかを組み合わせる方法で対処することになります。

SIONIIでは，

1) 背景キャラクター(MAX＝4)
2) 障害物(MAX＝6)
3) 敵キャラクター(MAX＝4)
4) 敵弾&敵ミサイル(MAX＝6)
5) 自機の弾(MAX＝3)
6) 自機のホーミングミサイル(MAX＝4)
7) エネルギーメーター(MAX＝2)

以上の７種類にキャラクターが分類されています。そして，プログラムでは，その種類ごとにMAX個数分のワークエリアを確保して管理しています。

キャラクターは，それぞれの種類ごとに番号を１から設定しています(図1－D)。このときは，別のキャラクターでも種類によって同じ番号のキャラクターが発生しますので，完全に種類ごとに表示ルーチン，出現ルーチン，キャラクター管理ルーチンなどを作成しなくてなりません。

同じキャラクター番号が発生する問題点を回避するためには，ゲーム中に登場するキャラクターをひとつのルーチンで処理できるように，キャラクター番号の設定を行う必要があります(図1－C)。この場合，それはそれで美しいプログラムなのですが，結局キャラクター番号をエリア分けして敵の種類を決めたほうが都合がよく，あまり変わらないといえます。

もちろん１から通し番号でキャラクター番号を設定すれば，エリア分けの必要がありません。しかし，ゲームデザインの段階でゲーム中に登場させるキャラクターをすべて考えているならともかく，たいていはボツになったりキャラクターを追加したりする場合が多いはずです。追加，削除する場合を考えると，なるたけ融通が利くような構造を考えるのが普通でしょう。

また，キャラクターの種類ごとに一括で処理できないような例外処理が，たくさん生まれる可能性があるのです。それを吸収するために条件分岐だらけの見づらいプログラムになってしまうのは，あまりにも情けないし効率的にも問題があります。

あちら立てればこちらが立たずという具合に，この管理の仕方によってルーチンの組み方がまったく違ってくるため，一概にはどれが決定的な方法とはいえません。つまり，作りたいゲームによって方式を変えるしかないのです。しかし，個人的にはキャラクターの種類でワークエリア，キャラクター番号を分割して管理する（キャラクター番号の重複をなくす）ほうが，ゲームの世界では都合がいいんじゃないかな，と思っています。

専用インタプリタ

次に知っておいて便利なアルゴリズムとして，専用インタプリタというものがあります。これは最低限のコマンド解釈部と実行部から成り立つインタプリタです。もちろん最低限というからには，エラーチェックなどというものがありません。不必要なデータ，誤ったコマンドでもおかまいなしにプログラムは実行していきます。つまり，ユーザーが完璧なデータを用意しておかなくては使いものにならないし，対話性もへったくれもありません。

では，なぜこのようなわかりにくく，使い勝手も悪い専用インタプリタを使用するのでしょうか。さっさと結論をいってしまうと使用する目的としては，サブルーチンのまとめと呼び出しの簡略化ができるためなのです。

具体的にいうと，敵キャラクターをA→B→Cの座標へ連続的に移動させ，C座標にきたときにはアニメーションをさせるとします。もちろん移動させるためのサブルーチンは用意されているとします。単純に考えると図2－Aのようなルーチンになるはずです。そこで，専用インタプリタを用意してプログラムを作成すると図2－Bのように簡略化されます。見比べればどちらが簡単かひと目でわかるはずです。なんだか○光お茶の間ショッピングのような説明ですが，実際に使って便利なのですからしようがありません。

コマンド設計例として，SIONIIで使われたベクトル移動ルーチンのコマンド表を掲載します(表２)。１命令は見事にコマンド番号とデータ部分，という具合に簡略化されているのがわかるでしょう。普段，X-

図1　キャラクター管理の例

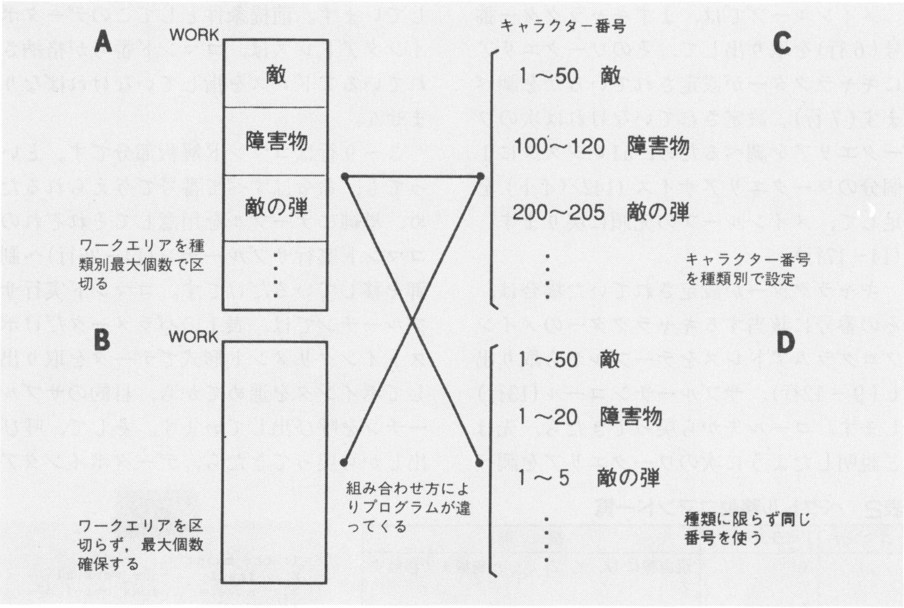
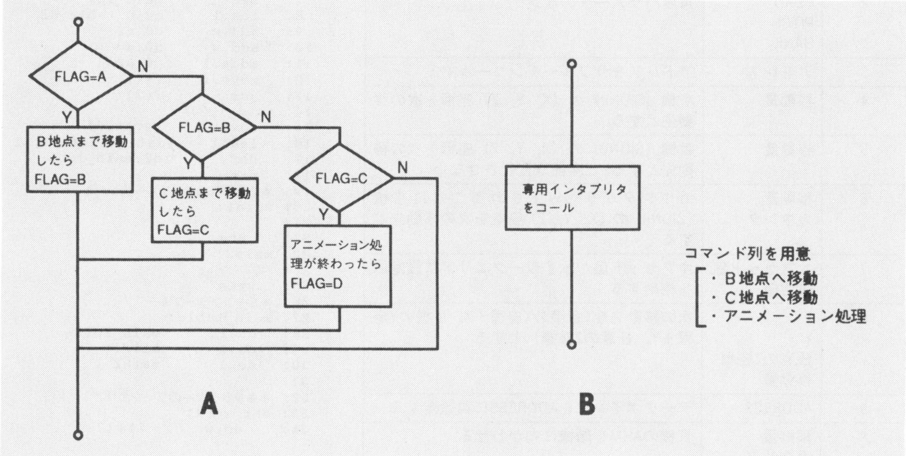


図2　アルゴリズムによるメインルーチンの違い



BASICという汎用インタプリタを使っている人にとっては,「こんなものは言語じゃない」と思うかもしれませんが, これだけでもある目的を果たすため立派に機能するのです。

コードに落とす

それでは, あまり抽象的な言葉ばかりで説明してもわかりづらいでしょうから, いままで説明したことをX68000のアセンブリ言語に落としてみましょう。

試しにキャラクターを管理するためのメインルーチンの骨格を書いてみます (リスト1)。

最初の初期設定部分 (1〜2行) にあるchr_workはキャラクターのデータが格納されているワークエリアのアドレスを指し, #max_chrは確保してあるワークエリアの個数を表しています。それぞれレジスタにセットしてからメインループに入ります。

メインループでは, まずキャラクター番号(6行)を取り出して, そのワークエリアにキャラクターが設定されているかを調べます(7行)。設定されていなければ次のワークエリアを調べるため, a1レジスタに1個分のワークエリアサイズ (142バイト) を足して, メインループの先頭に戻ります(14〜17行)。

キャラクターが設定されていた場合は, その番号に該当するキャラクターのメインプログラムアドレスをテーブルから取り出し(9〜12行), サブルーチンコール(13行)します。コール先から戻ってきたら, 先ほど説明したように次のワークエリアを調べる準備をして, メインループの先頭に戻ります(14〜17行)。

注意する点としては, ジャンプテーブルの28行目にあるアドレスデータです。これはサブルーチンコール(13行)のすぐあとのアドレスを指していて, 単独で見ただけだと意味のないデータと思えます。実際にこのアドレスを参照することはありません。しかし, キャラクター番号は1から始まっているために, 0番目のメインルーチンのアドレスとして登録しておく必要があるのです。別に4バイト分NOPで埋めてもいいのですが, なんとなく気分の問題で意味のないアドレスを指定しています。

このような調子で専用インタプリタのプログラムを書いてみましょう。1回呼び出されるたびにコマンドをひとつずつ実行し, 終了コマンド ($FFFF) で実行をやめるものを考えます (リスト2)。

まず, 2行でコマンドデータ列が記述されているデータポインタアドレスを取り出しています。前提条件としてこのデータポインタアドレスは, コマンド番号が格納されているアドレスを指していなければなりません。

3〜9行はコマンド解釈部分です。といっても, 命令はすべて番号で与えられるため, 単純にテーブルを用意してそれぞれのコマンド実行サブルーチン(14〜26行)へ制御を移しているだけです。コマンド実行サブルーチンでは, 表1のパラメータだけポストインクリメント形式でデータを取り出してポインタを進めてから, 目的のサブルーチンを呼び出しています。そして, 呼び出しから戻ってきたら, データポインタアドレスをワークに格納 (10行) して終了します。

構造は非常に単純なので簡単に理解できたでしょう。大切なのは, これらを発展させゲームに役立つものを設計することです。基本設計を怠ったため, SIONIIではえらい目に遭いました。この点だけはくどいといわれようとも力説しちゃいます。

ベクトル移動

3Dのベクトル移動を説明する前に, 2Dでのベクトル移動を復習してみましょう。

まず, A座標(0, 0)からB座標(8, 5)まで移動することを考えます。滑らかに移動させることを考えると, 図3のようにキャラクターは移動していくでしょう。この図をじっくり見ていると, X座標がいくつか更新されたときに, Y座標を1だけ更新していることに気づくと思います。逆にY座標が1更新されるには, X座標をいくつか更新したあとでなければならない, ともいえます。

で, このX座標をいくつ更新する (MX) かを求める計算は,

MX=((BX−AX)+余り)/(BY−AY)

のようになります。余りは前回の計算で発生した余りを使用します。この計算式で求めたMXをX方向のカウンタにし, MX=0になったらX座標の更新とY座標の更新を同時に行えばいいのです。終了判定はいろ

表2 ベクトル移動コマンド一覧

コマンド	パラメータ	機能
1	X Y Z 移動量	現座標に (X, Y, Z) 足した座標まで移動する
2	HEAD PITCH BANK	角度パラメータの設定
3	アドレス	アドレスをサブルーチンコールする
4	移動量	自機 (SIONII) の (X, Y, Z) 座標を次の移動先とする
5	移動量	自機 (SIONII) の (X, Y, Z) 座標を次の移動先とする。Z座標は変化させない
6	移動量 カウンタ	カウンタがリセットされるごとに自機 (SIONII) の (X, Y, Z) 座標を次の移動先とする
7	オフセット値 設定値	オフセット値が示すワークエリアに設定値を格納する
8	X Y 任意のZ座標 移動量	次の移動先を(自機のX座標+X, 自機のY座標+Y, 任意のZ座標) とする
9	ADDRESS	データポインタをADDRESSに再設定する
A	移動量 カウンタ	自機のAHMを敵機に向かわせる

リスト1

```
 1: chr_main:
 2:   lea.l      chr_work,a1
 3:   move.w     #max_chr-1,d7
 4: main_loop:
 5:   move.l   d7,-(sp)
 6:   move.w     d0,(a1)
 7:   beq        main_lp2
 8:   lea.l      main_table,a2
 9:   add.w      d0,d0
10:   add.w      d0,d0
11:   adda.l     d0,a2
12:   move.l     (a2).,a3
13:   jsr        (a3)
14: main_lp2:
15:   move.l     (sp)+,d7
16:   lea.l    #data_size(a1),a1
17:   dbf        d7,main_loop
18:   rts
19: *それぞれのメインルーチン
20: main1:
21:          :
22:    rts
23: main2:
24:          :
25:    rts
26: *ジャンプテーブル
27: main_table:
28:   dc.l       main_lp2
29:   dc.l       main1
30:   dc.l       main2
31:         :
32: *キャラクターのワークエリア
33: chr_work:
34:     dc.w       71*12
```

リスト2

```
 1: command:
 2:   move.l    (data_addr),a2
 3:   move.w   (a2)+,d0
 4:   bmi      command_ret
 5:   lea.l     command_table,a3
 6:   add.w     d0,d0
 7:   add.w     d0,d0
 8:   lea.l     (a3,d0),a4
 9:   jsr      (a4)
10:   move.l   a2,(data_addr)
11: command_ret:
12:   rts
13: *コマンド実行ルーチン
14: act0:
15:   move.w   (a2)+,d0
16:   bsr      act0_sub
17:   rts
18: act1:
19:     move.l   (a2)+,d0
20:     bsr      act1_sub
21:   rts
22: act2:
23:   move.w   (a2)+,d0
24:   move.w   (a2)+,d1
25:   bsr      act2_sub
26:   rts
27: *ジャンプテーブル
28: command_table:
29:   dc.l     act0_sub
30:   dc.l     act1_sub
31:   dc.l     act2_sub
32: *コマンドデータ列
33: command_data:
34:   dc.w     00
35:   dc.w     100
36:   dc.w     01
37:   dc.l     512
38:   dc.w     02
39:   dc.w     $ff
40:   dc.w     1554
41:   dc.w     $ffff
42: *データポインタワークエリア
43:   dc.l     command_data
```

いろな方法がありますが，いちばんわかりやすいと思われるのは，X方向の増分(BX－AX)をカウンタとしてワークエリアに保存しておき，X方向が更新されるたびに－1していき0になったら移動をやめればいいことになります。

ちなみに，この終了判定をはしょって計算を続けていくとA地点を始点にしてB地点を通る半直線になります。このアルゴリズムをフローチャートで書いたものが図4なので，説明だけでピンとこなかった人は，このフローチャートに従ってトレースしてみてください。

実際には，移動させる直線の傾きによって場合分けが必要となり，プログラムは多少複雑になります。

移動増分を考える

以上の説明は，X,Yともに移動増分＝1を前提としてきました。次に，任意の移動増分を考慮した場合，つまりキャラクターの移動に速度変化を付けることを考えます。

移動増分＝1のときには，X方向に1ずつ移動していきMX分移動が行われたら，Y方向に1の移動量を行うものでした。今度は任意の移動増分を考慮した場合，X方向にDX分の移動量が発生したら，Y方向にはどれだけの移動量(DY)が発生するかを計算しなくてはなりません。

図3の例で，X方向にDX＝2ずつ移動を行うとY方向の座標は直線とX座標の交点から求められます。で，Y方向の移動量(DY)はというと，

DDX＝(BX－AX)/DX

DY＝(DDX＋余り)/(BY－AY)

で計算できます。結局はほぼ同じような式になりますが，これをプログラムで書くと多少違ってきます(図5)。

なお，DDXを別の式で書いてあるのは，結果が定数で出てくるので，一度計算をすませておけば再度計算をする必要がないためです。

3Dでのお話

それでは，以上のアルゴリズムを3Dに転用することを考えます。2Dで示される移動ベクトルは，X,Y方向に分解できX,Y方向の移動をうまく行うことで，直線の移動ができました。3Dの場合は，移動ベクトルがX,Y,Z方向に分解され多少複雑に感じるかもしれません。しかし，実際にはこの3D空間にある3平面（X-Y，Y-Z，X-Z）のうち2平面に対して，2Dで行ったような直線移動のアルゴリズムを適用してやれば実現することができるのです。

この2平面を選択する方法は，極めて単純でX,Y,Z方向に分解されたベクトルのうち，絶対値がいちばん大きなものを基準として2平面を決定します。たとえば，V＝(30,50,－30)という移動ベクトルがあったとしたら，Y軸を基準にしてX-Y平面，Y-Z平面を選択するのです（図6）。

この方法の利点のひとつとして，前項で述べた線分移動アルゴリズムで傾きを考慮する必要がなくなる点があります。というよりも基準座標軸を判定した段階で，傾きを考慮したことになるためです。

そうして，基準座標軸に対する移動量の計算と，それ以外の座標軸に対する移動量の計算を繰り返すことにより，3Dでもベクトル移動ができるようになります。

図3　移動経路

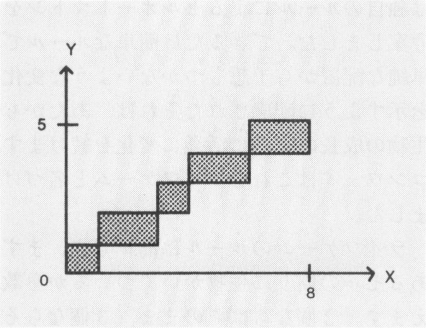
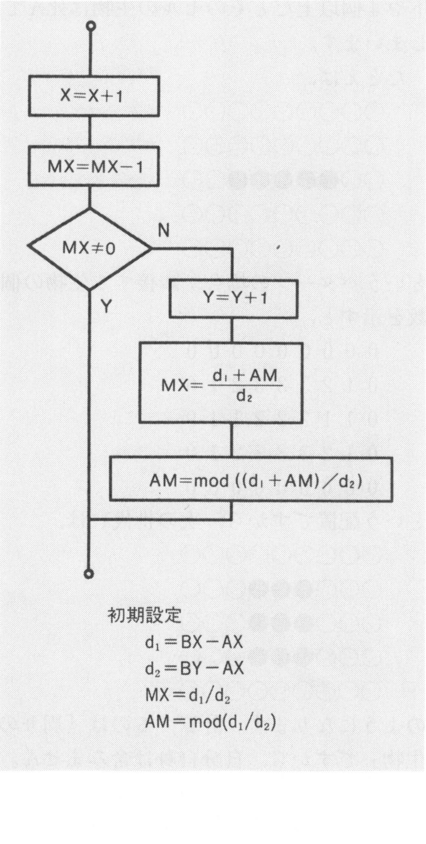


図4　ベクトル移動　その1



いつか会えるといいね

以上でSIONIIを通して，僕がいいたかったことをなんとか書き連ねてきました。ほとんど反省文とか，始末書を書いてる気分でしたが，自分のやってきたことを見直すことでいろいろな面が見えてきて，結構面白いものだなあと感じています。

ところで，最近はぼちぼちと次回作の構想を練り始めました。しかし，考える端から壁にぶち当たって全然はかどっていません。今度はサンプルプログラムなどという逃げを使わず，正々堂々とひとつの作品として皆さんに評価してもらえるモノを目指します。いつになるかはわかりませんが，ちょっとだけ期待してもいいかもしれませんよ。

では，またいつか会えるといいですね。

図6　3Dベクトルの基準ベクトルを見る

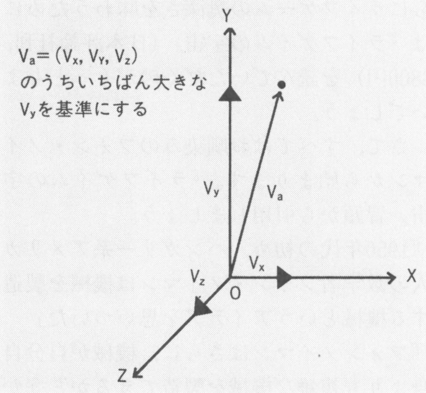
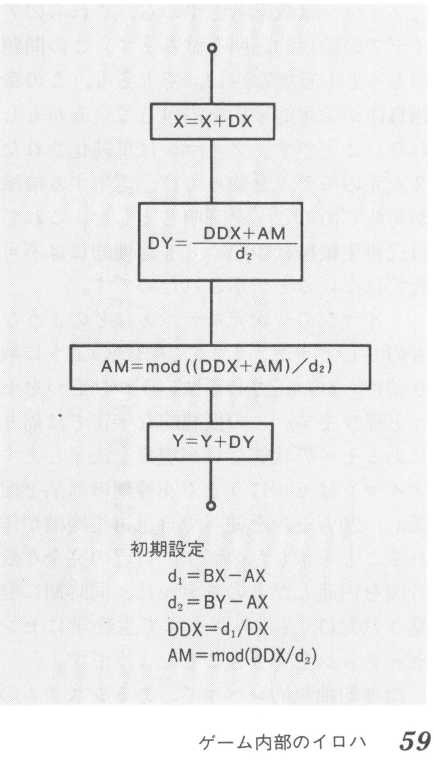


図5　ベクトル移動　その2



超高速ライフゲーム

LIFE110.X

Ishikawa Junji 石川 淳二

パソコンの草創紀より親しまれた「ライフゲーム」。LIFE110.Xはメモリの広さを生かしたコーディングによる，とことん速いライフゲームです。パソコンレベルを超えた大規模なセルオートマトンの展開も可能。これこそライフゲームの決定版ともいえるでしょう。

ライフゲームとはなにか?

ライフゲームとはコンウェイの考案した2次元セルオートマトンの一種で……と解説しても「セルオートマトンとはなにか?」から始めなければならなくなりそうなので，かいつまんで簡単に解説しておきます。さらにライフゲームの奥深さを味わうためには『ライフゲイムの宇宙』(日本評論社刊，3800円)を読んでいただくのがもっともよいでしょう。

さて，すべてはお馴染みのフォン・ノイマンから始まります。『ライフゲイムの宇宙』冒頭から引用しましょう。

「1950年代の初め，ハンガリー系アメリカ人の数学者フォン・ノイマンは機械を製造する機械というアイデアを思いついた」

「フォンノイマンはさらに，機械が自分自身よりも複雑な機械を製造できるかどうかも考えた」

ノイマンは数学者ですから，これらのアイデアの論理的証明を試みます。この問題のもっとも重要な点は，そもそも，この命題自体が論理的矛盾を内包しているかもしれないことです。ノイマンは単純化された2次元のモデルを使って自己再生する機械が可能であることを証明しました。これで，自己再生機械は少なくとも論理的には不可能ではないことが示されたのです。

ノイマンの2次元モデルとはどのようなものでしょうか。ここで方眼紙のように敷き詰められた正方形領域の1つひとつをセルと呼びます。この仮想的な宇宙では周りにあるセルの状態だけが現象を決定します。ノイマンはセルにうまく28種類の部品を配置し，20万セルを使って自己再生機械が作れることを示したのです。自己の完全な設計図を内蔵したそのモデルは，同時期に発見されたDNAとあいまって生物学にセンセーションを巻き起こしたようです。

論理的抽象的レベルで，あるシステムの動作を記述したものをオートマトンと呼びます。ノイマンのようにセル配置を使ったものをセルオートマトン，「セル状宇宙での自動機械」と呼ぶのです。

ライフゲームの特徴

1970年，ケンブリッジ大学のコンウェイは独自のルールによるセルオートマトンを考案しました。できるだけ簡単なルールで単純な配置から予想もつかないような変化を示すように配慮されたそれは，あたかも生物の成長のように活発に変化を続けます。コンウェイはこれをライフゲームと名づけました。

ライフゲームのルールは簡単です。まず，あるセルの周りに生物がいくついるかを数えます。2個ならばそのまま，3個ならそのセルに新しい生命が発生します。1個以下や4個以上だとそのセルの生物は死んでしまいます。

たとえば，

○○○○○○○○○
○○○○○○○○○
○○●●●●●○○
○○○○○○○○○
○○○○○○○○○

というパターンの場合，隣接する生物の個数を示すと，

0 0 0 0 0 0 0 0 0
0 1 2 3 3 3 2 1 0
0 1 1 2 2 2 1 1 0
0 1 2 3 3 3 2 1 0
0 0 0 0 0 0 0 0 0

という配置ですから，次の世代には，

○○○○○○○○○
○○○●●●○○○
○○○●●●○○○
○○○●●●○○○
○○○○○○○○○

のようになります。計算するのは「周りの生物」ですから，自分自身は含みません。

以下，

○○○○●○○○○
○○○●○●○○○
○○●○○○●○○
○○○●○●○○○
○○○○●○○○○

○○○○●○○○○
○○○●●●○○○
○○●●○●●○○
○○○●●●○○○
○○○○●○○○○

○○○●●●○○○
○○●○○○●○○
○○●○○○●○○
○○●○○○●○○
○○○●●●○○○

○○○○●○○○○
○○○●●●○○○
○○●○●○●○○
○●●●○●●●○
○○●○●○●○○
○○○●●●○○○
○○○○●○○○○

○○○○●○○○○
○○○○●○○○○
○○○○●○○○○
○○○○○○○○○
●●●○○○●●●
○○○○○○○○○
○○○○●○○○○
○○○○●○○○○
○○○○●○○○○

……のように変化していきます。最後のパターンは縦横の繰り返しになり，このパターンの進化は停止します。

もともとはオセロゲームのようなボード上でシミュレートされていたようですが，コンピュータの普及とともにディスプレイ

上のドットで生物を表し，変化を楽しめるようになってきました。特にマイクロコンピュータが現れてからはライフゲームはマイコンの定番アイテムとして普及していったのです。

その後，コンピュータの使用によりライフゲームは飛躍的に研究が進みました。よく現れるパターンや面白い変化をするパターンには固有の名前がつけられています。特に有名なもののひとつにRペントミノがあります。

ペントミノとはドミノから派生した言葉で５つの正方形を組み合わせてできる形を意味します。４つならテトロミノ，５つならペントミノ，以下ヘクソミノ，ヘプトミノ，オクトミノと続きます。ペントミノには12個の種類があり，Rペントミノはそのうちの，

　□□
□□
　□

という形状の組み合わせを意味します。このように初期画面を配置してライフゲームを始めると，非常に変化に富んだ展開を示すことが知られています。当初は無限に成長するパターンではないかとも思われていたようですが，コンピュータが発展するにつれ，1103世代後に安定することが確認されました。

●グライダー

ライフゲームの面白さを象徴するようなものがこのグライダーです。グライダーは世代が進むと斜め方向に移動する物体です。移動物体は総称して宇宙船と呼ばれていますが，グライダーはもっとも単純な宇宙船でもあります。

○●○
○○●
●●●

上記のような５個の物体を配置するとアラ不思議。４世代を周期として変化を繰り返しながら少しずつ移動していきます。LIFE110で実行するとピュンと飛んでいってしまうので注意してください。

●軽量級宇宙船

グライダー以外にもさまざまな形態の移動物体が発見されています。グライダーの次に簡単なのが軽量級宇宙船と呼ばれるものです。これは次のような配置になっています。

○○○●○
○○○○●
●○○○●
○●●●●

当然，軽量級以外にも中量級，重量級などが存在します。グライダー同士をうまくぶつけると宇宙船を作ることもできます。

●振動子

一定の周期で元の姿に戻る物体を振動子といいます。

●●●

というブリンカーがもっとも単純なものですが，なかには複雑な変化を繰り返して元に戻る見事なパターンもあります。以前，付録ディスクに収録されていたSXLIFE.Xのサンプルをご覧になった方はわかると思います。

●さまざまな固定物体

外界から刺激を与えられない限り，永遠に変化しない物体を固定物体と呼びます。

ライフゲームをやると，

●●
●●

といったブロックは非常にたくさん発生します。ほかにも，

○●●○
●○○●
○●●○

のような六角形の蜂の巣もよく見られます。固定物体はライフゲームでもっとも多く見られるパターンです。

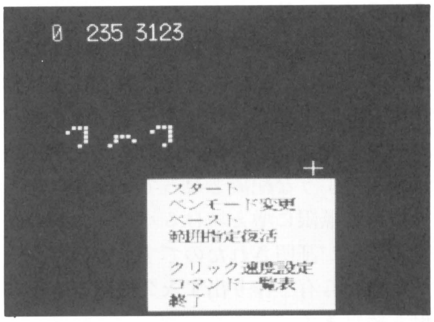
LIFE110.Xの使い方

さて，このようなライフゲームのルールをX68000でプログラムしたものが６月号の付録ディスクに収録されていたLIFE110.Xです。起動してみたものの，使い方がわからずに終わった人もいるかと思いますので簡単に使い方を解説しておきましょう。

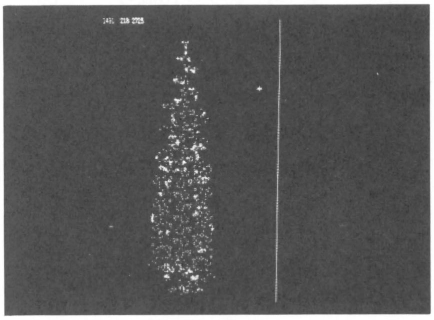
基本操作はヘルプキーを押してもらえればわかるはずですが，マウスの操作だけでパターンをセットするのではなくキーボードを併用することをおすすめします。

デフォルトの画面モードでは細かい操作には不向きですので，パターンをセットするときにはＪキーで画面モードを低解像度に変更するとよいでしょう。

基本操作はマウスで位置を指示し，Ｂキーでセットスペースバーでリセットになります（もちろんマウスで点を打ってもかまいませんが)。マウスの右クリックでメニューが出ます。カーソルモード変更とは，マウスでドットのセット/リセットを行うか，カット&ペーストを行うかを選択するものです。キーボードでセット/リセットを行い，マウスでカット&ペーストという設定にしておくと使いやすいと思います。



シュシュポッポ列車の初期配置


成長中のシュシュポッポ列車

大きなパターンの場合はＪキー（画面モードのキー）かUNDOキーでカーソル位置を画面の中央にすることができます。カーソル位置によって自動的にスクロールしたりしますが，自分で明示的にスクロール設定するほうがわかりやすいでしょう。

さて，このように初期位置として適当な場所に物体を置きます。あとは画面モードを切り換えて，マウスの右ボタンでメニューを出し，実行を選択します（あるいはキーボードのＧキー)。あとはじっと眺めるだけです。

永遠に成長するパターン

実際にライフゲームをやってみるとわかりますが，どんなに複雑に物体を配置していてもいずれは固定物体と振動子だけの世界，いわば進化の終焉を迎えます。

さて，コンウェイは考えました。

「無限に成長していくようなパターンは存在するだろうか？」

そして無限に成長するパターンをみつけるか，それが存在しないことを証明した者に50ドルの賞金をかけたのです。

コンウェイ自身は無限成長のパターンは発見できませんでしたが，もし存在するとしたらどのような動作をするかを推測しました。ひとつは振動子から周期的にグライダーが発生する「グライダー銃」，もうひとつは移動物体の一種で移動しながら軌跡に固定物体のゴミを残していくもの，すなわち「シュシュポッポ列車」です。

●グライダー銃

そして，ゴスパーを中心としたMITの学生グループはシャトルと呼ばれる往復移動物体２個の干渉でグライダーを生成するパターンを発見しました。このグライダー銃は図１のような配置となっています。

これで無限に増え続けるパターンが存在することが証明されたのです。このパターンは無から有を作り出す永久機関のように働き続けます。

同グループは，さらに13機のグライダーを衝突させてグライダー銃を作るという離れ業にも成功しています。

●シュシュポッポ列車

もうひとつの無限成長パターン，シュシュポッポ列車も存在が確認されました。それは図２のようなパターンとなります。シュシュポッポ列車は安定状態になるまで非常に巨大な領域を必要としますのでパソコンなどで再現するのは困難です。しかし，LIFE110.Xはメモリが増設されている場合ならばシュシュポッポ列車をほぼ再現することができます。メモリが増設されていない場合でも途中までの基本的な部分は確認することができます。

まず，Ｅコマンドで環境設定をします。カーソルキーの上下で矢印を動かし，左右で数値を増減します。スタックは768程度，エリアサイズは4Mバイト以上なら4096，２Mバイトなら取れるだけ最大に取っておいてください。

次に図２のように物体を配置します。終わったらMコマンドで保存するか，コピーバッファに入れておくといいでしょう。

あとは実行ですが，非常に大きなパターンですので，24kHzの使用できるディスプレイの場合はＰキーで画面を広く取っておいてください。LIFE110.Xでは左右がつながった構造になっているため途中で飛び出したグライダーにパターンが破壊されることがあります。そこで，

第2400世代くらいに右にひとつ
第2600世代くらいに左右にひとつずつ
第2700世代くらいに左にひとつ

飛び出してくるグライダーを取り除いてください。

5533世代を経てこのパターンは安定成長に入ります。そこまでの実行時間は10MHzで約30分くらいかかります。

このような永久成長パターンは安定すると一定の割合で画面のドット数を増加させます。しかし，グライダーでグライダー銃を作るアイデアを発展させて，世代が進むごとに加速的に増殖していくというパターンも開発されています。

３・４ライフゲーム

ライフゲームの生成規則を少し変えたものが３・４ライフゲームです。その名のとおり，周りの個体数が２個以下で消滅，３個と４個で生成，５個以上で消滅という規則となっています。

このようなルールだと固定物体がほとんど現れないという，独特の展開を示します。3･4ライフゲームでもいくつかの有名なパターンが存在しますが，なかでも，

　　　□□
　　□□
　□□

という階段状のヘクソミノについては，無限に増殖するものなのか，それとも有限の時間で安定するものなのか確認されていません。大型コンピュータを使ってなお解決されていない問題でしょうから，簡単にはいかないと思いますが誰か挑戦してみませんか？

さらに広がる宇宙

面白い動作をするグライダーという物体の存在は，ライフゲームの世界を大きく広げています。グライダーのやり取りを信号に見立てればデジタル回路を構成できます。グライダー銃の巧妙な配置でAND，ORといった基本的な論理回路が構成できることも示されています。

コンピュータのCPUというのは論理回路とメモリの集合体です。普通のCPUはトランジスタを使った論理回路（TTL）で構成されています。たとえば68000CPUの場合は68000個のトランジスタで作られているといわれています。１個の論理回路はだいたい３，４個のトランジスタから構成されます。つまり極論すれば，数十万個のグライダー銃を使えばセル空間上に68000CPUを再現できるかもしれないわけです。たかがライフゲームされどライフゲームといったところでしょうか。

＊　　＊　　＊

ライフゲームは「ゲーム」とはいっても実行されたときにはプレイヤーというものは存在しません。ただ規則どおりに画面が変化していくのを眺めるだけです。しかし，パターンがどのように変化するかを予測したり，自分の希望どおりの行動をするパターンをみつけたりといった部分は，ある種のパズル要素を持っています。そういった意味ではまさに知的遊戯といえるでしょう。

ミクロに見るとライフゲームの世界は仮想的なものであり，現実の宇宙とは限りなく大きな隔たりがあります。しかしマクロな視野に立つと別の宇宙でありながら，なにか似通った部分も現れています。皆さんもLIFE110.Xでライフゲームの深淵に触れてみてください。

図１　グライダー銃

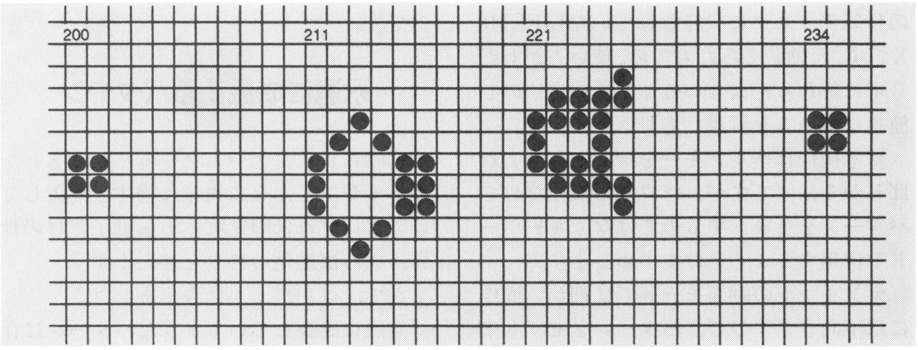
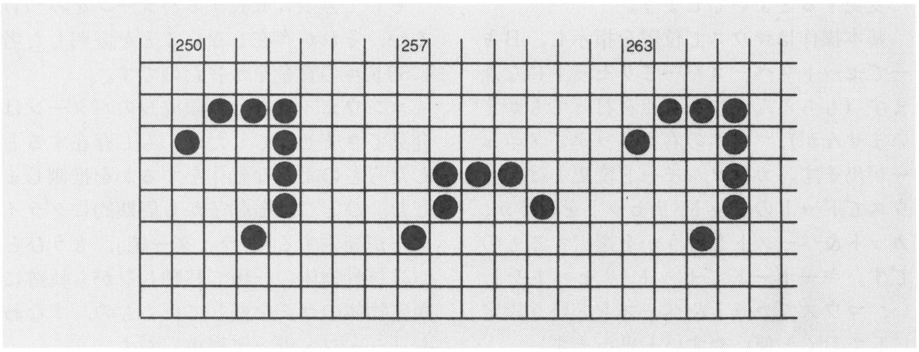


図２　シュシュポッポ列車



連載のすべて

(後編　1986−1991)

1986年

ビジネスソフトへの対応（増田　亨）
1986年1月号〜2月号，全2回
Microsoftの表計算ソフト「Multiplan」を使って，表計算とはどういうものなのか，どのようなメリットがあるのかを実例を交えながら解説したもの。当時はビジネスソフトがあまり普及しておらず,「Multiplan」が表集計ソフト市場を支配していた。

LOGOふたつの顔（向原あゆむ）
1986年1月号〜6月号，全5回
X1turbo用に発売されたLOGOを使った入門講座。現在では，LOGOとはなんなのか知っている人のほうが少ないかもしれない。子供用のオモチャのブロックの一種と勘違いされそうな名前だが(それはLEGO)，簡単な操作で扱えるタートルグラフィック(命令で亀を動かすと，その軌跡に線が引かれる)と，Lispのようなリスト処理を特徴としている。タイトルの「ふたつの顔」とは，この二面性のことをいったもの。

Prolog-85入門（高橋　明）
1986年2月号〜4月号，全3回
1985年12月号のTHE SENTINELで掲載された，S-OS用のProlog処理系であるProlog-85の入門講座。三段論法を逆に使って推論するPrologのプログラミングを，論理学の基礎から人工知能の入口まで解説している。プログラム例が決して多くないPrologだけに，毎回掲載されるプログラムは格好の教材となったことだろう。

掟破りへの挑戦（magiFORTH）
（山田伸一郎）
1986年3月号〜9月号，全6回
1986年3月号のTHE SENTINELで発表されたFORTH処理系の入門講座。スタックを使ってすべての処理を行う，という独特の作法をもつFORTHの魅力が語られた。FORTHはいわば仮想CPUのエミュレーションを行うプログラミング言語であり，そのCPUの仕組みを知ることが使いこなすための第一歩となる。FORTH処理系の内側を覗いてみようという人にもお勧め。

パソコン/ビデオユーザの映像処理入門
（文　秀則）
1986年5月号〜7月号，全3回
X1/turbo用に発売されたカラーイメージボードの能力と魅力を紹介する連載。カラーイメージボードはビデオなどの画像をわずか0.8秒（X1turboを使用した場合）という高速で取り込むもの。連載はソフトウェアとハードウェアの両方からカラーイメージボードに迫っており，取り込み画像の質をより向上させる方法，モザイク処理などの特殊効果の実現について触れている。

PC-1600Kの世界（河森　卓）
1986年7月号〜9月号，全3回
漢字を表示できるシャープ初のポケコン(?)PC-1600Kの機能と魅力を紹介する連載。PC-1600Kは本体のみならず，プリンタやディスクを付加したシステムもコンパクトで持ち運びできる。最終回ではプロッタプリンタを利用して，フラクタル図形の表示に挑戦している。

TURBO PASCALの世界（後藤貴行）
1986年7月号〜9月号，全3回
フルスクリーンエディタ，デバッガ内蔵のPASCALコンパイラであるTURBO PASCALの魅力を紹介する。CP/MのBDOSの機能を使ったグラフィックパッケージを用意し，その上でタートルグラフィックを行うなど，従来のPASCAL入門らしからぬアプローチが面白い。

Between The Lines（勝本　信）
1986年9月号〜1988年9月号，全22回
パソコン界の世相を斬るエッセイとしては，祝一平氏の「皿までどーぞ」が人気だったが，異なる筆者の異なる視点による連載は新鮮な印象を与えてくれる。

FuzzyBASIC料理法（瀧山　孝）
1986年10月号〜1987年1月号，全4回
1986年9月号のTHE SENTINELで発表された，S-OS用の構造化BASIC言語，FuzzyBASICの入門講座。従来のBASICがもっていない構造化命令の解説を中心に，BASIC処理系の内部構造まで丁寧に解説された。プログラミング言語作成法入門としても利用できる。

パソコン立体学"実践"講座（青木　実）
1986年10月号〜1987年1月号，全4回
赤青眼鏡を使った立体視から液晶シャッター方式に至るまで，立体視の原理を豊富な実例とともに解説した連載。テレビ画像をうまくデジタイズして立体視用のデータを起こしたり，X1/turbo用に発売された立体映像セットを使ってそれを眺めるなど実践的な連載となった。最終回では立体アニメプログラムが掲載されている。

知能機械概論（有田隆也）
1985年12月号〜現在
コンピュータに知能をもたせるためにはどうすればいいのか，という命題を中心に，ソフトウェアの話からハードウェアの話まで広く話題を提供する長期連載。コンピュータの研究者たちの最新の流行や話題など，パソコン誌にはめずらしい情報が提供されるエッセイである。

1987年

BASICリレー連載 プログラミング実況中継
1987年5月号〜1988年1月号，全9回
Oh!MZの筆者連持ち回りで行われたBASIC言語の連載。単一の筆者が基礎から解説する従来の連載とは異なり，それぞれにユニークなプログラムを提供し解説が加えられている。文法を学ぶものというよりは，プログラム作りの基礎，姿勢，方針，ノウハウといったものの解説が行われた。

BASICで数学と遊ぶ（八十　勉）
1987年5月号〜11月号，全7回
数学の先生による数学の連載。2次関数，双曲線関数などといっても，式を眺めて解析するのとグラフにして眺めるのとでは大違い。肩ヒジ張ったものではなく，パソコンとBASICを使って，グラフィカルな数学の世界に遊んでみようという趣旨のものであった。

X68000あなたの知らない世界
1987年7月号〜1988年8月号，全13回

1986年11月号の衝撃の発表以来，感動と，呆然と，自失と，期待のなかで待ち続けたX68000の発売。新製品ということで不足しがちなさまざまな情報をサポートするため，用意されたのがこの連載。内容はシステム，アプリケーションの多岐にわたっており，毎日を夢の中で過ごした蜜月の香りが漂っている。君は若さを失っていないか！

X68000BASIC入門（中森　章）
1987年8月号～1988年８月号，全13回

従来のBASICとはまったく異なる文法をもっている，X68000のX-BASIC。どちらかというとC言語に似た構造をしていて，従来のBASICで猛者と呼ばれた人もとまどった。当時，関数を中心とした新しいプログラミングスタイルに慣れている人はほとんどいなかったのだ。Oh!MZきっての言語通である中森氏によって，優しいX-BASICの入門連載が始まった。豊富なサンプルプログラムで愛された。

Oh!MZ(X) LIVE in
1987年10月号～現在

X1/turboに新たに用意されたFM音源に対応すべく，祝一平氏によってBASIC対応MMLが完成された。また，X68000は最初から８重和音のFM音源を搭載している。パソコンで高品質の音楽を楽しむ時代の到来である。この記事では読者からの投稿を中心に，各機種用の音楽演奏プログラムを掲載している。

SHORT ACCESS
1987年12月号～1988年11月号，全５回

アイデアと経験に築き上げられたプログラムの芸術ショートプログラム。数々の機能を盛り込んで，肥大化を続ける市販アプリケーションには真似のできない，研ぎ澄まされた一撃が身上だ。そんな投稿ショートプログラムを掲載した。

実用(？)オブジェクト指向のゲームプログラミング（浜口　勇）
1987年12月号～1988年８月号，全８回

オブジェクト指向は，なにもSmalltalkやC++の専売特許ではない。それは言語ではなく，むしろ本質的にはプログラミングへのアプローチの問題だ。上記の言語はそのアプローチを手助けするための機能をもっているにすぎない。この連載ではアセンブラを使ってオブジェクト指向を実践し，ゲームを作り上げていく。

人類タコ科図鑑（祝　一平）
1987年12月号～1988年７月号，全８回

名作「皿までどーぞ」復活の要望に応えて開始された，祝一平氏によるエッセイ。X68000の発表で，時代の進歩が予想よりも早かったことに対応して，満開2号の仕様が発表されるというひとコマもあった。満開2号の発売時期は満開1号よりも早いという設定だ。貿易摩擦真っ只中という時節柄，日米関係，日本人とアメリカ人といった点から問題を浮き上がらせたエッセイが多い。

1988年

Z80マシン語ゲーム工房（村田敏幸）
1988年８月号～1989年２月号，全７回

横スクロールシューティングゲームを制作しながら，Z80マシン語でのプログラミングの方法やコツを解説していくという，ゲーム作りによるマシン語講座。X1，MZなどのZ80という共通の土台を押さえながらも，各機種のハードに依存する部分も解説して生かしている。

C調言語講座PRO-68K（祝　一平）
1988年７月号～1990年６月号，全22回

文法はさておき，とりあえずCを使ってみてC言語を理解する，ということを前提に始められた祝一平氏によるC言語入門講座。著者が著者だけに，いーかげんにやっているように見えても，さすがに要所要所は押さえていた。対象はC compiler PRO-68Kで，B.W.カーニハン，D.M.リッチー著の「プログラミング言語C」(通称，K＆R)がサブテキストとして指定されていた。

OS9/X68000入門
1988年11月号～1989年５月号，全6回

1988年の暮れに，X68000にもうひとつのOSである「OS-9」が登場した。マルチタスクなど，このOSの特徴，オペレーティングの作法をさまざまなライターが紹介，さらにOS-9上のC言語である「C＆プロフェッショナルパッケージ」にも言及している。

われら電脳遊戯民
1988年８月号～1989年７月号，全12回

機種にこだわらず，ファミコンソフトなども取り上げて，ゲームの世界を考察する。筆者は毎月回り持ちで，各人のゲームに対する思い入れ，取り組み方，そして，ゲームとほかのメディア，現実社会との関わりが窺い知れた。

1989年

X68000マシン語プログラミング（村田敏幸）
1989年3月号～現在

X68000を対象にしたマシン語プログラミング入門。Human68kがどのようにメモリやファイル，プログラムを管理しているのかといった情報を含めて，デバイスドライバの作成方法や常駐プログラムの作成方法まで解説されている。前半の基礎編は，「X68000マシン語プログラミング 入門編」として単行本にまとめられた。X68000マシン語プログラマの必読書として人気が高い。

MZ-2500 MIDI入門（中田啓明）
1989年6月号～8月号，全3回

X1/turbo，X68000と次々と整備されるMIDI環境の中で，すっかり取り残されてしまっている観があるMZシリーズ。この状況を憂えた筆者が，MZ-2500用に開発したMIDIボードとそのコントロールプログラムを分載のかたちで紹介したもの。ボード制作のハードウェア編と，MIDIドライバ＋シーケンサプログラムのソフトウェア編，そして鍵盤表示プログラムのアプリケーション編に分かれている

マシン語カクテル in Z80's Bar
1989年7月号～現在

X1/turboシリーズを中心にした，Z80マシン語プログラミング講座。喫茶店(バー？)にやってくる登場人物の会話形式で話が進んでいく面白い形態をとっているのが特徴。掛け合い漫才のようなやりとりもあり楽しめる。現在の内容は，アルゴリズムを主体とした，マシン語のプログラミング方法の解説が中心。

X-BASICプログラミング調理実習（泉　大介）
1989年7月号～1991年1月号，全17回

X-BASIC入門講座の第2弾。マウスで演奏するギターや，エレベータのシミュレーションといった一風変わったサンプルプログラムが掲載されている。なかでもファイル入門の総集編として掲載されたカード型データベースは，手軽に手を加えることのできるデータベースプログラムとして人気があった。このプログラムはコンパイルされ，VS2対応のカード型データベースとして付録ディスクに収録されている。

DōGA・CGアニメーション講座（プロジェクトチームDōGA）
1989年7月号～1992年３月号，全22回

X68000でCGを使ったアニメーションが作成できる，と人気のDōGA・CGAシステム。その構成からモデリング，モーションデザインの方法，データ作成のアドバイスなどを制作者側がガイドする入門・活用講座。DōGA情報発信地としての役目も果たした。

MZ-2500用グラフィックエディタ作成講座（本橋　純）
1989年7月号～11月号，全5回

MZ-2500のもつ256色同時表示機能を生

かしたグラフィックエディタの作成講座。機能ごとの分載方式をとっており，毎回少しずつ機能を入力していけばグラフィックエディタが完成するようになっている。マルチウィンドウ，マスク機能つき，ソフトフォーカスに色調変換機能などが盛り込まれた力作。

(で)のショートプロぱーてぃ（古村　聡）
1989年8月号～現在

復活版SHORT ACCESSともいうべき，読者の投稿コーナー。時代とともに，プログラミング指向のものからアイデア一発感性指向のものへと投稿プログラミングは変化してきているが，高機能の市販プログラムが豊富に提供される現在ではそれも無理からぬこと。思わず笑えるプログラムだって大歓迎。楽しければいいじゃないか。

1990年

X-OVER NIGHT（高原秀己）
1990年6月号～現在

毎月，筆者の周りで起こった出来事，ふとしたことから湧いて出た疑問，不満などを書き連ねていく。やはりパソコン関連の話題が多いが，一般的な事柄なども時折取り上げられ，現代社会をさまざまな視点で見つめる。

大人のためのX68000（荻窪　圭）
1990年9月号～現在

X68000でいちばん充実しているアプリケーションは？　そう，ゲームだ。毎月のTHE SOFTOUCHを見ていても，ゲーム以外のプログラムのレビューが載ることはあまりない。それも，チンチンジャラジャラの派手なゲームが大半だ。オヂサンはこういうゲームを見ると疲れてしまう。ゲームなら腰を据えて取りかかれるもの，ついでに仕事も愛機X68000でできるならうれしい。そんなユーザーのために荻窪圭氏が書き下ろす連載だ。

ようこそここへC言語（中森　章）
1990年10月号～1991年12月号，全14回

プログラミング言語の生き字引，中森章氏によるC言語入門講座。豊富な実例とやさしい解説で，読者を一気にC言語ワールドへと導く。XC，GCCをターゲットに書かれているため，X68000ユーザーにとっては貴重なC言語の入門連載である。先頃，単行本にまとめられた。

PASCALプログラミングへの招待
（藤井義巳・藤木健士）
1990年6月号～1991年3月号，全7回

創刊8周年記念PRO-68Kと名づけられたOh!X第1号付録ディスクには，Pure PASCALコンパイラが収録された。そのPASCALを誌面でフォローすべく始まったのがこの入門講座。PASCALはC言語が全盛になったいまでも，プログラミング教育用の言語として多用されているが，これで学校の宿題もこわくない。自宅で心いくまでデバッグすることが可能となったのだ。

清水和人流プログラミング道場
1990年9月号～1991年4月号，全4回

ひさびさの連載となった，清水和人氏のアマチュアプログラマ支援講座。技法や構造といった考えはひとまず置いて，暇に任せてつらつらと気まぐれにプログラムを組む。そして，気にくわないところを見つけては改良を重ねるという方法を実践し，読者に勧めている。サンプルプログラムも“演歌の歌詞自動発生”，懐かしの“テーブルテニスゲーム”とユニークなものが取り上げられている。

シミュレーションプログラミング入門
（華門真人）
1990年12月号～1991年9月号，全6回

コンピュータの大きな需要のひとつであるシミュレーション。重力のシミュレーションプログラムなどはめずらしくないが，身近な出来事を取り上げ，本格的に社会現象をパソコンでシミュレートしてみようという試みは貴重だ。筆者多忙のため，惜しまれつつ最終回を迎えた連載である。

ハードウェア工作入門（三沢和彦）
1990年6月号～現在

コンピュータの高機能化にともなって，パソコンはシロウトがハードウェアに手を出せないブラックボックスと化している。それでも，パソコンを動かしているナマの情報に触れたいという潜在的な欲求は強い。丁寧な解説は，ハードウェアを勉強してみようという人にもぴったりだ。

INTEGRAL X1（亀田雅彦）
1990年6月号～1991年2月号，全7回

創刊8周年PRO-68Kに掲載された，X1用コマンドシェルシミュレータ“INTEGRAL X1”の拡張機能やデバッグ情報などを掲載。X1上でMS-DOSやX68000のHuman68kのような環境を得ることを目指して作られているだけに，MS-DOSのファイルの読み書きも可能で，操作はMS-DOSに準拠していた。最初はX1turboシリーズのみ対応であったが，連載の途中でX1もサポートされた。

PC-E500テーブルトークRPGサポートシステム（松井　信）
1990年8月号～10月号，全3回

ひと昔前のパソコン並みの機能をもったポケットコンピュータ「PC-E500」を，テーブルトークRPGを楽しむのに利用する。戦闘時の勝利判定といった面倒な計算をさせるためにゲームマスター支援ツールCSTを制作，誌面で紹介した。

1991年

吾輩はX68000である（泉　大介）
1991年4月号～現在

1990年11月号の特集「理科系のGAME REVIEW」で登場した，「吾輩はパソコンである」がもとになって始まった連載である。タイトルから察せられるとおり，夏目漱石の「吾輩は猫である」をモチーフにしており，主人公である“X68000”から見た視点で，自らの機能を解析，また，活用したりしている。この連載では主にデバッガを使用し，1つひとつの命令，動作を確認しながら，話が進められる。

よいこのSX-WINDOW講座（中森　章）
1991年4月号～現在

謹賀新年PRO-68KにSX-WINDOWの技術資料，ツール類，Cライブラリが収録された。それらを踏まえたうえで，C言語を使って具体的なプログラミングの方法を解説していく。サンプルプログラムも多数掲載。情報の公開，そしてSX-WINDOW ver.2.0の登場により，“SXプログラマ”はますます増えていくことであろう。

Creative Computer Music入門（瀧　康史）
1991年10月号～現在

音楽演奏プログラムを打ち込むのもいいけれど，自分で作曲というのもまた楽し，ということで始まったコンピュータによる作曲入門の連載。基本的な音楽理論や作曲のコツを，ゲームミュージックの楽譜などを交えながら説明している。

響子 in CGわ～るど（寺尾響子）
1991年6月号～現在

第3回アマチュアCGAコンテストに「HEART」で入選した，主婦兼イラストレーターの寺尾響子さんによるCGとイメージストーリーを毎月掲載している。X68000が2台，そして，トランスピュータという環境でCGは作られている。

ANOTHER CG WORLD（寺尾響子）
1991年10月号号～現在

上記の「響子 in CGわ～るど」で掲載されるCGを，ユニークなマンガで解説。解説といっても，難解な理論が展開されるわけではなく，あくまで描く過程でのコツやむずかしいところが紹介されるのがミソ。

みんな準備はいいか？

プロジェクトチーム DōGA かまた ゆたか
MAX田口

先月号の付録ディスクは「DōGA・CGAシステム ver.2.50＆お試しディスク」でした。CGAシステムはともかく，お試しディスクは遊びつくされたことでしょう。来月からはいよいよ新システムを使った連載が始まりますが，その前に……。

先月号の付録ディスクはいかがでしたか？　お試しシステムでかっこいいアニメーションはできましたか？　マニュアルもないうちからCGAシステムを展開して，“わけわからん！”と叫んでいるんじゃないでしょうね？　さて，本格的なCGA講座の連載はマニュアルがちゃんと行き届く来月からとして，今月はプレ連載として，マニュアルに関するお知らせと，お試しシステムのちょっと高度な使い方を紹介します。

──昔々，あるところに牛を飼っているおじいさんがおりました。日本一おいしい牛肉を目指し，何年間も努力し，やっと納得できる牛となりました。そして，その肉を村人たちに食べてもらおうと，みんなに配りました。ところが村人は，料理するのは面倒だといって，生肉のままかじりつき，食えたもんじゃないと，吐き捨てました。おじいさんは，泣くに泣けぬ思いでした──

いったい何の話だって？　つまり，マニュアルなしにCGAシステムを使わないでねってことです。CGAシステムとは，プログラムだけを指しているのではありません。マニュアルがあって，初めて意味をなすのです。どんなにうまい肉でも，料理しなければ食べられないのと同じ。マニュアルもないのに強引に使って，「使いものにならない」といわれるのだけは避けたいのです。

世の中のマニュアルなんて，どっちみち読まない，何かわからないことがあったとき調べるだけ……というものだと私自身思っています。しかし，このCGAシステムのマニュアルは違います。“マニュアル”って言い方が悪いのかもしれませんね。“CGA制作攻略本”，あるいは“DōGA教の教典”とでも呼びましょうか（オイオイ）。

各プログラムの機能がズラーッと並んでいるようなところ（機能一覧編）もありますが，それは全体の３割以下です。それでは残る７割はというと，別表のようになっています。総ページ数がなんと800ページ。そういってもピンとこないという方は，本屋に行ってマンガの「月刊ジャンプ」や「月刊マガジン」を手に取ってください。サイズ的にちょうど同じです。前回は“置けばひとり立ちできるマニュアル”といわれていましたが，今回は“殴れば人を殺せるマニュアル”と呼ばれています。

もちろん，マニュアルはページ数が多ければよいというものではありません。その点もご安心を。今回，本当にできるかぎりの努力をし，内容の濃い，読んで面白いマニュアルを目指しました。極端な話，CGAに興味のない人が読んでも楽しめるのではないでしょうか。

ということで，マニュアルはCGAシステムのプログラム以上の自信作です。ひとりでも多くの方に読んでいただきたいので，ぜひ申し込んでください。締め切りはもう間近です（７月31日）。

申し込みには先月号の専用振替用紙をご利用ください。詳しい申し込み方法も先月号に書いてあります。先月号は持っていないという方も若干名いらっしゃるかもしれませんが，その場合は付録ディスクも持っていないでしょう。残念ながら現在は，ディスク付きのマニュアル発送は行っておりませんので，そういった方は，ひととおりマニュアルの発送が終わったあとで別途対応したいと思います。詳しくは次号をご覧ください。

お試しシステムでひとひねり

そろそろお試しシステムにも飽きたころでしょうから，ちょっと工夫して，お試しシステムで作った複数のカットを一度に連続してアニメーションする方法について説明しましょう。うまく編集すれば，ちょっとした作品にもなりますよ。Human68kやエディタはある程度使えるものとして話を進めます。そうでない方は，Human68kのマニュアルか，CGAシステムのマニュアル「CGA大学/教養課程/パソコン基礎概論」で勉強してください。

1）　アニメーションデータを用意する

一度に再生できるアニメーションの量は画像の複雑さによって異なりますが，画質が普通の場合，メモリ１Mバイトあたり５～10秒です。ですから，まずご自分のメモリにあわせて，アニメーションをお試しシステムで作ってください。たとえば，メモリが２Mバイトなら，３秒のカットを４～６カット準備すればいいでしょう。

2）　データ名を変更する

さて，お試しシステムでは同じ物体を使用した場合，画像データ名が同じになってしまいます。つまり，「ワゴン」を選択した場合の画像データは「WAGON???.PIC」となります(???には３桁の数字が入る)。

一度にアニメーションさせるカットの中に，同じ物体を使ったものがあれば，画像ファイル名も同じになるので区別がつきません。そこで，Human68kのコマンドのREN(リネーム)を使って，ファイル名を変更します。ここでは，「WAGON???.PIC」を「WAGOA???.PIC」に変更する方法を解説します。なお，リネームする際には，必ずファイル名の文字数を同じにしてください。

まず，データディスクの「PICS」というディレクトリに移ります。

CD ¥PICS

そして，

REN WAGON???.PIC WAGOA???.PIC

とします。これで，ファイル名が「WAGOA」+「3桁の数字」+「.PIC」に変更されます。

このようにして，アニメーションさせる画像データの中に同じファイル名が存在しないようにしてください。

3) タイムチャートファイルを作る

次にタイムチャートファイルを書き換えます。タイムチャートファイルとは，アニメーションさせる順番を定義しているファイルです。

まず，最初のカットが入っているデータディスクを用意してください。そのディスクの¥PICSの中に，拡張子が「.TCH」になっているファイルがひとつだけあるはずです。それをエディタで書き換えます。

たとえば，「WAGON.TCH」の中身が，

```
.timechart
    WAGON [1-60]
.endchart
```

となっていたとします。これは，画像データを WAGON001.PIC，WAGON002.PIC，…… WAGON060.PIC まで順番にアニメーションするという意味です。アニメーションの速度は毎秒20フレームですから，3秒のアニメーションということになります。

そこで，このタイムチャートファイルを自分の用意した画像データにあわせて書き換えます。たとえば，

```
.timechart
    WAGOA [1-60]
    WAGOB [1-40]
    WAGOC [1-80]
.endchart
```

とすると，「WAGOA」という3秒のアニメーションに続いて，2秒間の「WAGOB」，4秒間の「WAGOC」を連続してアニメーションします。書き換えが終わったら，ファイルをセーブしてエディタを終了してください。

4) アニメーションの実行

さあ，これで準備は整いました。あとはアニメーションを再生するだけです。1カット目のデモディスクを0ドライブに入れて立ち上げます。先ほどのタイムチャートファイルならば，画像ファイルを順番に読み込み，「WAGOA 060.PIC」を読み込み終わると，画面に「can't open WAGOB 001.PIC」と表示されます。この表示が出たら，ディスクを2カット目のディスクと交換し，リターンキーを押します。すると，続けて2枚目の画像ファイルを読み込み始めます。あとは同じようにして，画面に「can't open ～」と表示されたら，順番にディスクを交換してください。すべて読み込み終わったら，アニメーションが始まります。どうですか。ちゃんと続けて一度に再生されていますか？

5) 仕上げる

ちょっと難しかったですか？　とりあえずつなぐことはできても，でたらめにつないだだけでは，なかなか作品にはなりません。カットのつなぎ方，カットの構成など，考えなければならないことはいっぱいあります。そのあたりの理論は「CGA大学/修士課程/映像理論概論」で勉強してもらうとして，とりあえず，試行錯誤でいろいろやってみてください。うまくいけば面白いものができるかもしれません。友達に見せて自慢してください。

どうでもいいですが，個人的には「物体：ワゴン，背景：道路，動き：アクロバット」が，どうみてもワゴンが何かに跳ね飛ばされているみたいで好きです。では，来月号から本格的に始まる連載をお楽しみに。

注）一度に再生するアニメーションの時間が長すぎる場合，「メモリが足りません。～が読み込めませんでした」と表示され，再生プログラムが終了することがあります。このような場合は，一度に再生するカットの数を減らして，もう一度チャレンジしてください。

マニュアル構成

ご使用になる前に	お題目です。作画待ち時間にでも読んでください。
CGA大学編	マニュアルを入手したら，ここから入ってください。過去の連載を参考に，大幅に加筆，修正しました。
─教養課程	パソコン，CGの超初心者のために，基礎知識を身につけます。
─専門課程	具体的な実習で，とりあえずCGAシステムを使えるようになりましょう。
─修士課程	作品制作のうえで知っておくべき事柄を，体系的に学びます。
─博士課程	より高度な表現を目指しています。ここを全部理解できる人は，そう多くはないでしょう。
機能一覧編	各プログラムの機能を網羅しています。
クイックマニュアル	各種データの事例集など，さっと調べたいような事項の抜粋で，作品制作時には重宝するでしょう。
おまけ	分冊時に自由に使う表紙，中扉集など。

ディスクトラブル

現在，先月号の付録ディスクに大きな問題点は発見されていません。細かな点では，以下のようなミスがありました。

○CGAシステムをハードディスクにインストールした場合，PESからCOMMAND.Xが呼べない（PES.DEFでA:¥BINに設定されているため）。

○BETA.Xがデバッグ中バージョンになっていた。2画面目を取り込んだとき，バスエラーが起こる。

○FFのdiv I で放物線を微分したときの値がおかしい。

○お試しシステムのREADME.DOCに誤字があった。

気にするほどのことじゃありませんね。間違って付録ディスクをフォーマットしちゃったなどのトラブルは，編集部のほうで対応してくれるそうです。

お詫び

第4回CGAコンテストのビデオの発送が，マニュアル制作のために大幅に遅れましたことをお詫びします。アマチュアとはいえ，もっとしっかりせねばと反省しております。

そういえば，DōGAの法人化について，いくつかご意見をいただきました。ありがとうございます。ここのところ忙しくて進展してませんが，財団法人案が急浮上しています。財団法人というのは，ある寄付金をもとに，営利を目的とせず，科学，文化，芸術などの公益を追求するための法人というのですから，DōGAにぴったりです。まさに，アマチュアのプロ(?)といえるでしょう。

しかし，手続き上，不特定多数からの寄付金でも可能か，寄付金の最低限度額を満たすか，大阪府知事の認可が得られるか，などの問題があります。大阪府の知事さんが冗談を理解してくれる人ならうれしいんだけどな（笑）。どなたか法律などに詳しい方，アドバイスください。

よいこのSX-WINDOW講座————————(第9回)

イメージを極める

Nakamori Akira 中森 章

SX-WINDOW上でのグラフィック解説の最後は「イメージ」についてです。グラフマンには，さまざまな形式のビットイメージデータとサポート関数があります。それらの特徴と使い分け方をまとめてみていきましょう。

はじめに

レクタングル，リージョンと，これまでグラフマンの扱う図形のデータ構造と，それを操作する関数について見てきました。今回はイメージを見ていきましょう。イメージとはドットの集合で任意の図形を表すものです。イメージを極めればグラフマンの扱う図形は全部征服したと思ってもいいのではないでしょうか。

イメージとは

パソコンのディスプレイの解像度を表す言葉として「何ドット×何ドット」というものがあります。これは，パソコンのディスプレイをドット（点）の集まりとみなして，縦方向と横方向にドットがいくつあるかを示す言葉です。この言葉からわかるように，パソコンのディスプレイとはドットが平面上に敷き詰められたものにほかなりません。

パソコンのディスプレイの上に映し出される図形は，これらの各ドットがどういう色をしているかによって決定されます。周りと違う色のドットはそこに点が描かれていると認識されます。そういったドットが隣り合って並んでいれば直線や曲線が描かれていると認識されます。たとえば，

○○○●○○○
○○●○●○○
○●○○○●○
●○○○○○●
●●●●●●●
●○○○○○●
●○○○○○●

という縦7ドット，横7ドットの平面にある●の集まりはアルファベットのAを表すことができます。

直線であろうと，曲線であろうと，パソコンのディスプレイに表示される図形は色違いのドットの集まりにすぎません。パソコンのディスレイに描かれる図形は，それがどのような形状のものでも，基本はドットの集まりとして考えることができます。そして，パソコンで図形を表示するためには，図形をドットの集まりとして扱えるような仕組みが不可欠となります。SX-WINDOWではドットの集まりとしての図形をイメージと呼び，それを操作する関数をグラフマンが提供しています。

グラフマンが扱うイメージには，大別すると，

ビットイメージ
レクタングルイメージ
プロットイメージ
G16イメージ

の4種類があります。これらは，どれもドットの集まりとして図形を表すという点では同じものですが，データ構造が少しずつ異なります。まずは，これらのイメージについて解説しましょう。また，イメージの分類を図1に示しておきます。

1) ビットイメージ

ビットイメージとは，ドットが敷き詰められた平面上に定義されるイメージのことです。この平面は概念上では無限の大きさを持っていればいいのですが，それではイメージをデータとして扱う場合に不便ですから横方向と縦方向に適当なドット数を与えて考えます。先に示した「A」というイメージは横方向，縦方向がともに7ドットの大きさを持った平面上に定義されていました。そのビットイメージは，次の，

○○○○●○○○○○○
○○○●○●○○○○○
○○●○○○●○○○○
○●○○○○○●○○○
○●●●●●●●○○○
○●○○○○○●○○○
○●○○○○○●○○○

というビットイメージや，

○○○○○○○
○○○○○○○
○○○●○○○
○○●○●○○
○●○○○●○
●○○○○○●
●●●●●●●
●○○○○○●
●○○○○○●
○○○○○○○
○○○○○○○

というビットイメージとどこが異なるのでしょうか。もちろん，ビットイメージとして見ればどれも同じものです。最初の「A」のイメージは，横7ドット，縦7ドットという，その図形（文字）を表現するための最小の領域（レクタングル）で示してあるだけなのです。つまり，ビットイメージを扱う場合，そのビットイメージが占める領域を一緒にして考えなければ，データの大きさを一意に決めることができません。

実例を見ればわかりますが，ビットイメージを引数として渡す関数では，同時にそのビットイメージの占める領域を示すレクタングルも引数で渡すようになっています。

さて，SX-WINDOWではビットイメージが定義される平面の性質に応じて，

テキストタイプ
グラフィックタイプ
GR2タイプ
GR3タイプ

という4種類のビットイメージをデータとして扱うことができるようになっています。以下にそれぞれの具体的なデータ構造について説明しましょう。

●テキストタイプのビットイメージ

これは，X68000のテキストVRAM（テキスト画面）上に表示されることを目的としたビットメージです。テキストVRAMは，横方向に連続する16ドットをひとまとめにして扱うようになっています。このため，テキストタイプのビットイメージは1ドッ

トを１ビットに対応させ，16ビットをひとまとめとしたワードデータの集まりで図形を定義するようになっています。

ひとつのワード内ではMSB（一番位の高いビット）が図形の左方向，LSB（一番位の低いビット）が図形の右方向に対応し，ワードとワードの間では（左から数えて）最初のものほど図形の左方向に対応します。たとえば，ビットイメージのある１行が，

○○●●○○○●○○●○○○○●●
●○●○●●○○○●●●●○○

であるとします。これをビットに対応させると，

0 0 1 1 0 0 0 1 0 0 1 0 0 0 0 1 1
1 0 1 0 1 1 0 0 0 1 1 1 1 0 0

となり，さらに16ビットずつにまとめると，

0 0 1 1 0 0 0 1 0 0 1 0 0 0 0 1，
1 1 0 1 0 1 1 0 0 0 1 1 1 1 0 0

となります。ワード内ではMSBがイメージの左，ワード間では先（左）にあるものがイメージの左なので，先に示したビットイメージの１行は16進数で，

3121_H，$D63C_H$

となるのです。いまは１行のドット数が16の倍数（32）になっていましたが，そうでない場合は右方向に無意味なビットを付加して１行のドット数が16の倍数になるようにする必要があります。

そして，この１行のビットイメージデータを，順番に，ビットイメージの縦方向のドット数と同じ数だけ並べるとテキストタイプのビットイメージデータができあがるというわけです。ただし，こうして定義されるビットイメージはテキストVRAMの１ページ（プレーン）分にすぎません。

X68000のテキストVRAMは最大４ページまでのビットイメージの重ね合わせで図形や文字を表示します。各ページの対応するビットがテキストVRAM上での同一ドットを表し，それぞれのビットの値に従って表示される色（パレットコード）が決定されるようになっています。

パレットコードの値は，

第４ページのドットの値×８
＋第３ページのドットの値×４
＋第２ページのドットの値×２
＋第１ページのドットの値×１

によって計算されます。ドットの値は０か１ですから，最大16種類の色を表示することができるのです（図２）。

ビットイメージもテキストVRAMの各ページに対応して用意する必要があります。つまり，テキストタイプのビットイメージの最終的な形式は，それぞれのページに対応するビットイメージのデータを第１ページから順番に，使用するテキストVRAMのページ数と同じだけ並べたものとなります。

たとえば，使用するテキストVRAMを全部で２ページとした場合，先に示した「A」という図形のビットイメージは，

```
short image [] = {
  /* 1プレーン用 */
    0b0001000000000000,
    0b0010100000000000,
    0b0100010000000000,
    0b1000001000000000,
    0b1111111000000000,
    0b1000001000000000,
    0b1000001000000000,
  /* 2プレーン用 */
    0b0001000000000000,
    0b0010100000000000,
    0b0100010000000000,
    0b1000001000000000,
    0b1111111000000000,
    0b1000001000000000,
    0b1000001000000000
};
```

というshort型の配列で表現できます[1]。このとき，配列の各要素の下位９ビットは１行のデータを16ビットの倍数にするために０を詰めてあります。また，イメージで●の部分のパレットコードを３，○の部分のパレットコードを０にしてあります。

1）0b～というのはＸＣでの拡張表現で２進数を表す。GCC（真里子版）では環境変数「真里子」の中にＡという文字があれば，この２進数の表現が使えるようになる（警告メッセージは出る）。

●グラフィックタイプのビットイメージ

これはX68000のグラフィックVRAM（グラフィック画面）に表示されることを目的としたビットイメージです。X68000のグラフィックVRAMは俗にパックドピクセルと呼ばれる方式で，16ビットのパレットコードを１ドットとして扱うようになっています。そこで，グラフィックタイプのビットイメージはテキストタイプのビットイメージとは異なり，グラフィックVRAMの１ドットを１ワードのパレットコードで表現します。この１ドットのデータを横方向のドット数だけ並べたものが１行のデータになります。そして，この１行のデータを縦方向のドット数と同じだけ並べたものがグラフィックタイプのビットイメージとなります。

なお，１行の中では最初（左）にあるデータほど，ビットイメージの左方向のデータに対応しています。

先ほどの「A」というイメージをグラフィックタイプのビットイメージで定義すると次のようになります。ここでは，イメージで●の部分のパレットコードを１，○の部分のパレットコードを０にしてあります。

```
short image [] = {
    0,0,0,1,0,0,0,
    0,0,1,0,1,0,0,
    0,1,0,0,0,1,0,
```

図１　イメージの分類

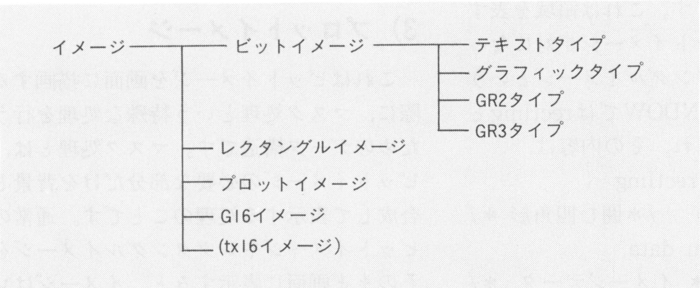
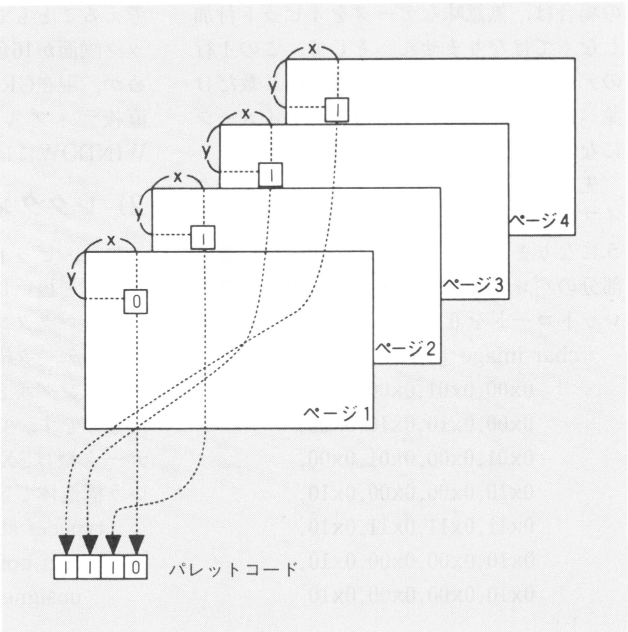


図２　テキストVRAMのドットの値とパレットコード



```
        1,0,0,0,0,0,1,
        1,1,1,1,1,1,1,
        1,0,0,0,0,0,1,
        1,0,0,0,0,0,1
    };
```

このように，グラフィックVRAMにはページという概念がないだけ，ビットイメージはすっきりとした形で定義できます。

ところで，SX-WINDOWでは，現在のところ，グラフィックタイプのビットイメージを直接パソコンのディスプレイに表示するための関数は用意されていません。1ドットを表すために1ワード（16ビット）が必要になるのはグラフィック画面が65536色表示モードの場合ですが，現在のSX-WINDOWのシステムは16色表示モード以外はまともにサポートされていないためでしょう。将来の拡張用といった感のあるデータ構造です。

●GR2タイプのビットイメージ

GR2タイプのビットイメージはグラフィックタイプのビットイメージの変形です。

1ドットを1ワードのパレットコードで表していたのを，4ビットに圧縮するという発想です。プログラムをするうえで4ビットという単位は扱いにくいので，さらにデータを2ドットずつ組み合わせて1バイト（8ビット）のデータとしたのがGR2タイプのビットイメージなのです。このとき，1バイトの上位4ビットが左側の点，下位4ビットが右側の点を表すように組み合わされます。

1行のデータはビットイメージの左から右の順で格納されます。2ドットずつの組み合わせですから横方向のドット数が奇数の場合は，無意味なデータを4ビット付加しなくてはなりません。そして，この1行のデータを縦方向のドット数と同じ数だけ並べたものがGR2タイプのビットイメージになります。

先ほどの「A」というイメージをGR2タイプのビットイメージで定義すると次のようになります。ここでは，イメージで●の部分のパレットコードを1，○の部分のパレットコードを0にしてあります。

```
    char image [] = {
        0x00,0x01,0x00,0x00,
        0x00,0x10,0x10,0x00,
        0x01,0x00,0x01,0x00,
        0x10,0x00,0x00,0x10,
        0x11,0x11,0x11,0x10,
        0x10,0x00,0x00,0x10,
        0x10,0x00,0x00,0x10
    };
```

1ドットが4ビットということは，16色を表せるということです。テキストVRAMは最大16色表示ですし，グラフィックVRAMも通常は16色表示になっています。したがって，GR2タイプのビットイメージは，テキスト画面とグラフィック画面の両方と相性のよいデータ構造になっています。

●GR3タイプのビットイメージ

これは，グラフィックタイプのビットイメージの8ビット版です。つまり，1ドットを1バイト（8ビット）のパレットコードで示すデータ構造です。1ドットを表すビット数の違い以外はグラフィックタイプのビットイメージとまったく同じです。

先ほどの「A」というイメージをGR3タイプのビットイメージで定義すると次のようになります。ここでは，イメージで●の部分のパレットコードを1，○の部分のパレットコードを0にしてあります。

```
    char image [] = {
        0,0,0,1,0,0,0,
        0,0,1,0,1,0,0,
        0,1,0,0,0,1,0,
        1,0,0,0,0,0,1,
        1,0,0,0,0,0,1,
        1,1,1,1,1,1,1,
        1,0,0,0,0,0,1,
        1,0,0,0,0,0,1
    };
```

グラフィックタイプのビットイメージのときとshortがcharに変わっただけですね。

GR3タイプのビットイメージでは1ドットが8ビットであるため，最大256色までを表示できます。これはグラフィック画面の256色モードに対応するビットイメージと考えることもできます。ただし，グラフィック画面が16色モードを基準にしているためか，現在GR3タイプのビットイメージを直接ディプスレイに表示する関数はSX-WINDOWには用意されていません。

2）レクタングルイメージ

先に，ビットイメージは領域を込みで扱わないと扱いにくいということを述べましたが，レクタングルイメージはまさにそのためのデータ構造です。これは領域を表すレクタングルとビットイメージが対になったものです。レクタングルイメージというデータ型はSX-WINDOWではrectImgという構造体で定義され，その内容は，

```
    typedef struct rectImg  {
        rect bounds;      /*囲む四角形 */
        unsigned char data;
                          /* イメージデータ  */
    } rectImg;
```

となっています。ビットイメージを表すデータの前に，領域を示すレクタングルが置かれています。このレクタングルは，各メンバの値を，

```
    left      ← 0
    top       ← 0
    right     ← 横方向のドット数
    bottom    ← 縦方向のドット数
```

で指定するのが普通です。

ところで，実際のプログラムでrectImgというデータ型を使用する場合，dataという構造体のメンバは可変長になるので少々工夫が必要です。たとえば，GR2タイプやGR3タイプのレクタングルイメージでは，

```
    typedef struct rectImg  {
        rect bounds;      /*囲む四角形*/
        char data [100];
                          /*イメージデータ*/
    } rectImg;
```

などと変形して使うべきでしょうし，テキストタイプやグラフィックタイプのレクタングルイメージではイメージデータがshort型になりますから，

```
    typedef struct rectImg  {
        rect bounds;      /*囲む四角形*/
        short data [100];
                          /*イメージデータ*/
    } rectImg;
```

として使うべきでしょう。このようにrectImgというデータ型は扱うビットイメージのイメージデータに応じて使い分ける必要があります[2)]。

また，rectというデータ型を示す構造体のメンバのすべてがshort型であることを考えると，テキストタイプやグラフィックタイプの場合はrectImg全体をshort型の配列と見なすことも可能でしょう（配列の第5要素からがイメージデータになる）。

2）ビットイメージやレクタングルイメージを引数とするSX-WINDOWの関数は，イメージをポインタの形式で与えることが多いので，rectImg型へのポインタへキャストしてやれば，イメージデータの実体はshort型の配列でもchar型の配列でもなんでもよい。

3）プロットイメージ

これはビットイメージを画面に描画する際に，マスク処理という特殊な処理を行うためのデータ構造です。マスク処理とは，ビットイメージの必要な部分だけを背景と合成して表示する処理のことです。通常のビットイメージやレクタングルイメージをそのまま画面に表示すると，イメージはレ

クタングルで指定される領域の単位に処理されるので，レクタングルで指定する部分の背景が書き潰されてしまいます。マスク処理はビットイメージの中で，本当に表示させたい図形以外の背景を書き潰さないように保護するための処理なのです（図3）。

現在，プロットイメージはテキストタイプのビットイメージに限定されています。プロットイメージでは通常のビットイメージの後ろに，マスク用にもう1ページ分のイメージデータが付加されます。このとき，使用するページ数が4より少なければ，必要なページ数分のデータの直後にマスクデータが続きます。マスクはビットイメージの中で背景と合成したい領域のドットに対応するマスクデータのビットを1にすることで指定します。たとえば，

```
short plot_image [] = {
/* 1プレーン用 */
  0b0001000000000000,
  0b0010100000000000,
  0b0100010000000000,
  0b1000001000000000,
  0b1111111000000000,
  0b1000001000000000,
  0b1000001000000000,
/* 2プレーン用 */
  0b0001000000000000,
  0b0010100000000000,
  0b0100010000000000,
  0b1000001000000000,
  0b1111111000000000,
  0b1000001000000000,
  0b1000001000000000,
/* マスク用 */
  0b0001000000000000,
  0b0011100000000000,
  0b0111110000000000,
  0b1111111000000000,
  0b1111111000000000,
  0b1000001000000000,
  0b1000001000000000
};
```

というプロットデータは「A」という図形において，枠線とそれに囲まれた△の部分だけを背景と合成するデータです。

なお，SX-WINDOWのドキュメントなどによると，プロットイメージは最大4ページ分のイメージデータの後ろに，マスクを付加できる（合計5ページ）ように記述されていますが，実際は3ページ分のイメージデータとマスクの組み合わせが最大の組み合わせになります。

SX-WINDOWのリソースとしてPAT3，PAT4と呼ばれるもの(パターンエディタ.Xで編集できるイメージの形式）があります。これは，それぞれ，2ページ分，3ページ分のビットイメージ(テキストタイプ)にマスクのビットイメージが付加されたプロットイメージです。したがって，パターンエディタ.Xを利用すればプロットイメージは簡単に作成することができます。

4）G16イメージ

SX-WINDOWのデータ構造を定義してあるsxdef.hというファイルを見るとG16イメージはtx16というイメージと混同されているようです（あるいはGR2とG16が混同されている)。G16イメージとはGR2タイプのレクタングルイメージにほかなりません。ちなみに，tx16とは，

```
typedef struct tx16 {
unsigned int self;       /*'TX16' */
int length;              /*バイト数*/
rect bounds;             /*囲む四角形*/
unsigned short palet [16];
                         /* パレット*/
unsigned char data;
                   /* イメージデータ */
} tx16;
```

によって定義される，パレット付きのテキストタイプのイメージ（おそらくはGR2)のようです。しかし，現在SX-WINDOWにはtx16というデータ型を扱える関数は存在していません。あとで出てくる，

GMDrawG16

という関数の第1引数はtx16型データへのポインタになっていますが，これはG16イメージ（GR2タイプのレクタングルイメージ）へのポインタでなければなりません。注意してください。

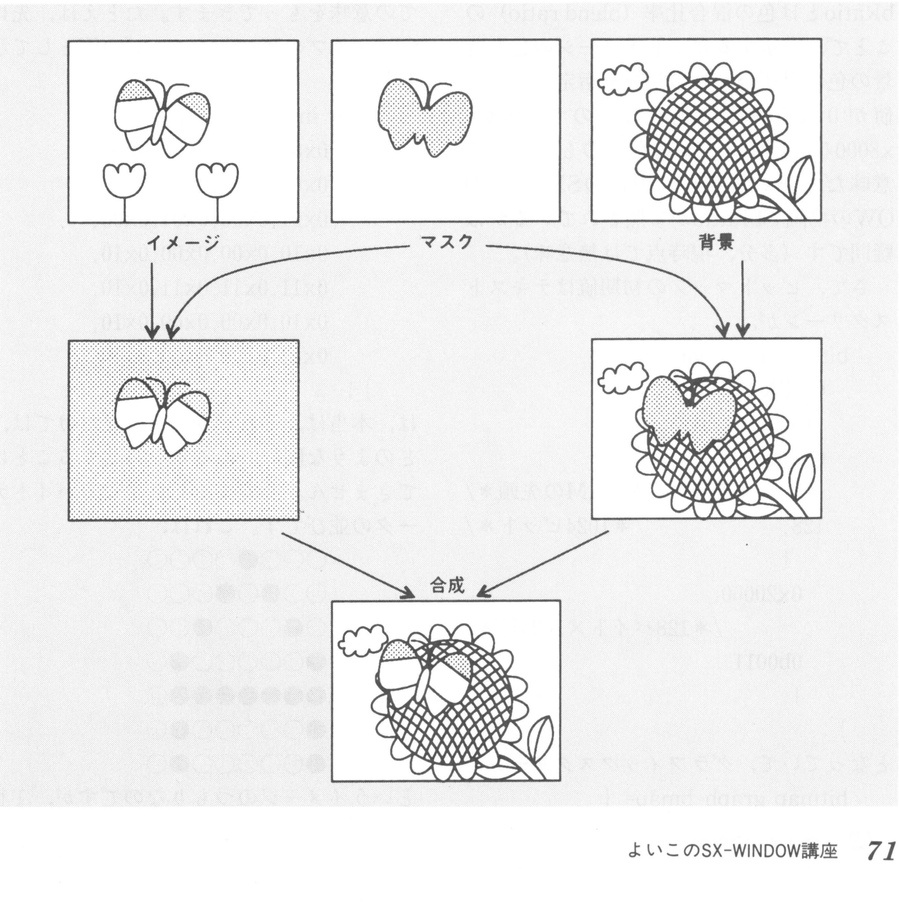
ビットマップとは

イメージの種類がわかったところで，次はビットマップについて説明しましょう。ビットイメージとはドットが敷き詰められた平面上に描かれた図形ですが，その平面の親玉（？）としてビットマップという概念が存在します。SX-WINDOWでは描画を行うための画面をスクリーンと呼んでいます。テキスト画面に対応するのがテキストスクリーン，グラフィック画面に対応するのがグラフィックスクリーンです。ビットマップとはこのスクリーンの構成を決定づける構造体であり，グラフマンがスクリーンに対して描画を行うときの情報を保持しています。

SX-WINDOWで使用するビットマップの構造体は次のように定義されています。

```
typedef struct bitmap {
  short bmKind;   /*画面の種類*/
```

図3　マスクの概念



```
    rect bmRect;  /*画面の大きさ*/
    int base;     /*ベースアドレス*/
    short line;
          /*横1ラインのバイト数*/
    union  {
      unsigned short bRatio;
                      /*混合比率*/
      TBM   tbm;
        /*テキスト画面固有の情報*/
    }  opt;
  }  bitmap;

 typedef struct TBM  {
    int  page;
            /*1ページのバイト数*/
    unsigned short aPage;
             /*アクセスページ数*/
  }  TBM;
```

これだけの情報があればスクリーンに一応描画を行うことができるのです。スクリーンとはX68000のハードウェア（VRAM）に直接対応する概念ですから，ハードウェアが異なればそのための描画情報は異なってきます。

bitmapという構造体の終わりのメンバがbRatioとtbm(page,aPage)に分かれているのは，テキストスクリーンとグラフィックスクリーンで異なる描画情報が必要なためで，前者がグラフィックスクリーン用，後者がテキストスクリーン用です。なお，bRatioとは色の混合比率（blend ratio）のことで，表示するビットイメージの色と背景の色の混ぜ合わせの割合を指定します。値が0なら背景の色そのもので，値が0x8000ならばイメージの色そのものという意味だと思いますが，現在のSX-WINDOWの描画でbRatioが参照されているかは疑問です（多分，現時点では無意味）。

さて，ビットマップの初期値はテキストスクリーンが，

```
    bitmap text_bmap= {
      G_TXT,
       {0,0,768,512},
      0xe00000,
            /*テキストVRAMの先頭*/
      128,              /*1024ビット*/
        {
        0x20000,
             /*128バイト×1024行*/
        0b0011
        }
    };
```

となっていて，グラフィックスクリーンが，

```
    bitmap graph_bmap= {
      G_GRP,
       {0,0,768,512},
      0xc00000,
        /*グラフィックVRAMの先頭*/
      2048,  /*2バイト×1024ドット*/
      0x8000
    };
```

となっています。画面の種類であるGTXT，G_GRPというのは表示できるビットマップの種類と読み変えて差し支えありません。G_TXTがテキストタイプのビットイメージ，G_GRPがグラフィックタイプのビットイメージに対応しています。

当然，GR2タイプ，GR3タイプのビットイメージを表示するためのスクリーンもあります。しかし，これらの形式に対応するハードウェアはX68000にはありませんから，これらに対応するスクリーンは仮想的なものになります。つまり，メモリ上に定義されるビットマップという概念です。メモリ上に定義される仮想的なビットマップをSX-WINDOWでは「ビッツ」と呼んでいますが，ここでは説明は省略します。詳しく知りたい方は追補版SX本を参照してください。

ただし，メモリ上の仮想的なビットマップという概念はビットイメージを扱ううえで非常に重要な概念です。ビットイメージとは，そのビットイメージのタイプと描画される領域が決定されて初めて，図形としての意味をもってきます。たとえば，先にGR2タイプのビットイメージの例として示した，

```
    char image [] = {
        0x00,0x01,0x00,0x00,
        0x00,0x10,0x10,0x00,
        0x01,0x00,0x01,0x00,
        0x10,0x00,0x00,0x10,
        0x11,0x11,0x11,0x10,
        0x10,0x00,0x00,0x10,
        0x10,0x00,0x00,0x10
    };
```

は，本当は，それが与えられただけでは，どのような図形であるかを特定することはできません。このままでは単なるバイトデータの並びです。これは，

○○○●○○○○
○○●○●○○○
○●○○○●○○
●○○○○○●○
●●●●●●●○
●○○○○○●○
●○○○○○●○

というイメージのつもりなのですが，1行のドット数を半分にした，

○○○●
○○○○
○○●○
●○○○
○●○○
○●○○
●○○○
○○●○
●●●●
●●●○
●○○○
○○●○
●○○○
○○●○

というビットイメージを表す場合でも，同じデータになってしまいます。つまり，与えられたデータが，ビットイメージのうち，テキストタイプなのか，グラフィックタイプなのか，GR2タイプなのか，あるいはGR3タイプなのかが指定され，何バイトが1行分のデータなのかが指定されなければビットイメージというものは定まらないのです。

このように，ビットイメージを特定するのに必要な，ビットイメージの種類，横1行分のバイト数などの情報を格納するデータ構造がビットマップであると考えることもできます。つまり，ビットイメージを定義する場合は，それが表示されるビットマップを暗黙のうちに決定していることになるのです。上に示したimageというビットイメージは「A」という図形を表すGR2タイプのイメージのつもりですから，それは，同時に，

```
    bitmap graph_bmap= {
      G_GR2,      /*イメージの種類*/
       {0,0,8,8},
           /*イメージを囲むドット数*/
      image,      /*イメージの先頭*/
      4,          /*1行のバイト数*/
      0x8000              /*適当な値*/
    };
```

という（仮想的な）ビットマップを定義していることになり，その上にビットイメージが表示されることを期待していることになるのです。

ところで，SX-WINDOWのウィンドウをオープンするときに指定するウィンドウのIDでグラフィックサポート（48，50）というものがあります。これらのIDでオープンしたウィンドウのグラフポートのビットマップがグラフィックタイプになると考える人がいるかもしれません。しかし，それは間違いで，ウィンドウがオープンされ

たときのビットマップは必ずテキストタイプになっています。グラフィックサポートというのはテキスト画面の背景が透明になっていて，グラフィック画面が透けて見えるようになっているだけです。グラフィック画面に図形を描くためには，あとで説明するGMSetBitmap関数やGMExgBitmap関数で，グラフポートにグラフィックタイプのビットイメージを設定する必要があります。

イメージを操作する関数

SX-WINDOWでイメージを扱う関数はグラフマンの管轄下にあります。ただし，イメージというのはパソコンのディスプレイの画面モードに依存する部分が多く，そのデータ構造もひとつではありません。SX-WINDOWで画面モードに関する部分は仕様的に不明確，あるいは発展途上と思われる部分が多く，それらをサポートする関数類も未整理のままといった感がぬぐえません。

イメージに関するグラフマンの関数も，それらの事情をもろに反映していると考えられます。ここではイメージを扱う関数のなかで基本的なものに的を絞って説明しておきます。

表1にイメージを扱う関数で基本的な関数の一覧を示します。順次解説しましょう。

●GMPutRImg

レクタングルイメージを指定した点から描画します。GMPutRImg関数が引数として取ることができるのはテキストタプのイメージだけです。それ以外のイメージではなにも描画されません。なお，イメージはかならずテキスト画面に描画されます。

●GMPutImg

ビットイメージをレクタングルで指定する位置に描画します。GMPutImg関数が引数として取ることができるのはテキストタイプのイメージだけです。それ以外のイメージではなにも描画されません。なお，イメージは必ずテキスト画面に描画されます。GMPutRImg関数との違いはレクタングルイメージを使用するか，領域を指定したビットイメージを使用するかだけです。

●GMDrawG16

G16イメージを指定した点から描画します。イメージは，現在のグラフポートのビットマップがテキストタイプならテキスト画面に，グラフィックタイプならグラフィック画面に描画されます。

sxlib.hのプロトタイプ宣言では，第1引数のデータ型はtx16型へのポインタとなっていますが，これは間違いです。

●GMTransImg

異なるスクリーンタイプ間でビットイメージをコピーします。つまり，テキストタイプ，グラフィックタイプ，GR2タイプ，GR3タイプのビットイメージを変換してテキスト画面やグラフィック画面に表示（描画）することができます。転送元と転送先の領域の大きさを変えることで，拡大・縮小をしながらの描画が可能です。

●GMCopy

同じスクリーンタイプ間でビットイメージをコピーして描画します。転送元と転送先の領域の大きさを変えることで，拡大・縮小をしながらの描画が可能です。また，コピーするときにコピーモードを指定することができます。これは，ペンモードと同じ意味を持ち，描画時に，ANDとかORといった，背景との演算を指定することができます。ペンモードに関してはSX本などを参考にしてください。

さらに，リージョンで範囲指定を行うことにより，転送先で実際に描画を行う範囲を指定することができます。範囲指定が不要なら，リージョンへのハンドルの代わりに0を指定します。どうやら，GMCopy関数はビットイメージを描画するためのもっとも基本的な関数のようです（GMPutImg，GMPlotImgなどの関数は最終的にはGMCopyを呼び出す？）。

●GMPlotImg

プロットイメージをレクタングルで指定する位置に描画します。プロットイメージはテキストタイプのものしか存在しません。なお，描画と同時にプロットモードを指定することができます。これは1バイトのデータで，上位4バイトで使用するテキスト画面のページ数，下位4ビットで描画の属性を指定します。ページ数は0から3で，0のときは2を指定するのと同じ意味があります。

SX本などの解説ではページ数として4を指定できるような記述もありますが，4以上の数を指定すると描画は行われません。下位4ビットには次のような意味があります。10以上の値を指定すると描画は行われません。

0…標準（そのまま描画）
1…反転（0の色反転）
2…ハイライト（強調）
3…ハイライト反転（2の色反転）
4…消去（背景色で描画）
5…消去（4と同じ）
6…網掛け
7…網掛け反転（6の色反転）
8…不可視（網掛けの1種？）
9…不可視反転（8の色反転）

これらの描画属性はSX-WINDOWのシステムがアイコンの表示などを変更するときに使用するものと予想されます。

ところで，1ページ分のプロットイメージは128×128ドット以内ということになっていますが，ビットイメージの1行のドット数が16の倍数でないと正しく描画されな

GMGetBitmapの修正

sxlib.aの中のGMGetBitmap関数は，現在のグラフポートのビットマップへのポインタを正しく返しません。そこでライブラリ関数を修正します。

```
        .xdef _GMGetBitmap
        .text
_GMGetBitmap:
        .dc.w $a1c7
        move.l a0,d0
        rts
```

というプログラムをA1C7.Sというファイル名で作成しましょう。その後，

as A1C7.S

のようにアセンブルしてA1C7.Oというオブジェクトファイルを作成し，

ar /u sxlib.a A1C7.O

によってライブラリを更新すれば終了です。

表1　イメージを扱う基本的な関数

関数名	機能
GMPuTRImg(rectImg*,point_t)	レクタングルイメージを描画する
GMPutImg(unsigned short*,rect*)	ビットイメージをレクタングルで指定する位置に描画する
GMDrawG16(tx16*,point_t)	G16タイプのイメージを描画する
GMTransImg(bitmap*,rect*,rect*,)	異なるスクリーンタイプ間でビットイメージをコピーする
GMCopy(bitmap*,rect*,rect*,int,region**)	同じスクリーンタイプ間でビットイメージをコピーする
GMPlotImg(unsigned short*,rect*,int)	プロットイメージを描画する
GMSetbitmap(bitmap*)	現在のグラフポートのビットマップを変更する
GMGetbitmap()	現在のグラフポートのビットマップを得る
GMExbitmap(bitmap*)	現在のグラフポートとビットマップを交換する

注（　）内は引数のデータ型を示す。

いようです。

●GMSetBitmap

現在のグラフポートのビットマップを指定したビットマップに変更します。

●GMGetBitmap( )

現在のグラフポートのビットマップを得ます。ただし，sxlib.aで供給されるGMGetBitmap関数にはバグがあるので，囲み記事を参照して修正しておいてください。

●GMExgBitmap

現在のグラフポートとのビットマップと指定したビットマップを交換します。

プログラムの例

それでは，イメージを扱う関数を使用したプログラムの例を示しましょう。説明することはあまりないので，一度に示します。

ただし，イメージを扱うプログラムでは，そのイメージデータの定義だけでかなりの行数を使ってしまうので，ここでは使用するイメージデータは同じものとし，そのファイルをほかのプログラムでインクルードして参照するようにしています。なお，今回使用したパターンは岡野哲也氏の作成したKo-Window上のKoNEKO2.win（version 1.60）のパターン（本来はX-Winodwのxnekoやonekoのパターン）を流用しています。

また，メインプログラムはすべて共通で，画面に描画を行う部分とヌルイベントの部分が異なるだけなので，メインプログラムと，それ以外のプログラムを別個に示して分割コンパイルを行うようにしています。

●リスト1…テキストタイプのビットイメージを定義してあるファイルです。32×32ドットのビットイメージがneko1, neko2というshort型配列に定義されています。テキスト画面のページ数は4面を使用するものとし，4ページ分のイメージデータが格納されています。また，最後のページのイメージデータはプロット時のマスクデータとしても使用できるようになっています。リスト1のプログラムはほかのプログラムではsx_pat.cというファイル名でインクルードされています。

●リスト2…グラフィックタイプのビットイメージを定義してあるファイルです。32×32ドットのビットイメージがneko_gr, neko_gr2, neko_gr3というchar型配列（neko_grはshort型）に定義されています。これらのイメージデータは，それぞれ，グラフィックタイプ，GR2タイプ，GR3タイプになっています。リスト2のプログラムはほかのプログラムからsx_pat2.cというファイル名でインクルードされています。

●リスト3…リスト4～リスト7のメインプログラムです。リスト3のプログラムはヌルイベント時に実行するIDLEという関数と，アップデートイベント時に実行する描画のためのDRAWという関数を別ファイルで定義することを期待しています。リスト4～リスト7のプログラムはIDLE関数とDRAW関数が定義されているだけです。したがって，リスト4～リスト7のプログラムはリスト3のプログラムをリンクしなければ正しく動作しません。

リスト3のプログラムではポップアップメニューでX68000のテキスト画面の各ページとグラフィック画面をクリアできるようにしてあります。本来ならSX-WINDOWのシステムが管理しているテキスト画面やグラフィック画面をクリアするのは反則なのですが，ウィンドウに描かれた図形がテキスト画面のものかグラフィック画面のものかを調べるため，あえてプログラムに入れてみました。ウィンドウ上の図形が，テキスト画面をクリアすることによって影響を受ければ，それはテキスト画面上に描かれたもの，グラフィック画面をクリアすることで消えれば，それはグラフィック画面に描かれたものと判断することができます。

●リスト4…GMPutRImg関数，GMPutImg関数，GMDrawG16関数を用いて描画を行うプログラムです。ビットマップを切り替えることにより，グラフィック画面とテキスト画面の両方に描画を試みています。GMPutRImg関数やGMPutImg関数はテキストタイプのみしかサポートしていないとの説明どおり，グラフィクタイプのビットイメージをグラフィック画面に描画しようとしてもなにも描画されません。また，テキスト画面とグラフィック画面で対比が取りやすいように，テキスト画面のパレットをグラフィック画面のパレットにコピーしてあります。

なお，リスト1やリスト2で定義したビットイメージはレクタングルイメージではないので，レクタングルイメージが必要な場合は，tempという配列にレクタングルの値とビットイメージをコピーしてレクタングルイメージを作り出しています。

リスト4の実行結果を写真1に示します。

●リスト5…GMTransTmg関数を用いていろいろなタイプのビットイメージをグラフィック画面とテキスト画面の両方に描画するプログラムです。GMTransImg関数はビットマップからビットマップへのコピーですから，ビットイメージから仮想的なビットマップを作り出す必要があります。リスト5のPutImg関数の最初のほうで，イメージのタイプに応じて仮想的なビットマップを作り出しているのがわかると思います。その仮想的なビットマップから現在のグラフポートのビットマップであるwinPtr->wGraph.bmapにコピーを行っています。このように，GMTransTmg関数を用いれば，すべてのタイプのビットイメージを現在のグラフポートのビットマップに描画することができるのです。

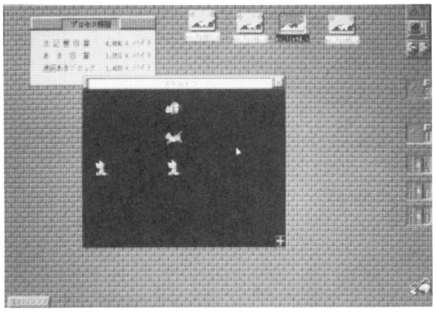


写真1　リスト4の実行例

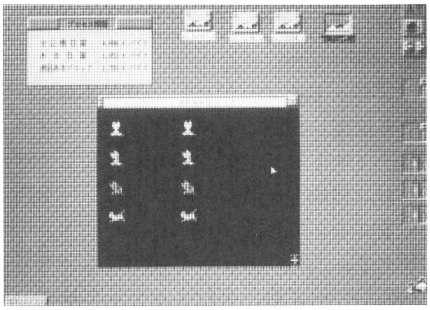


写真2　リスト5の実行例

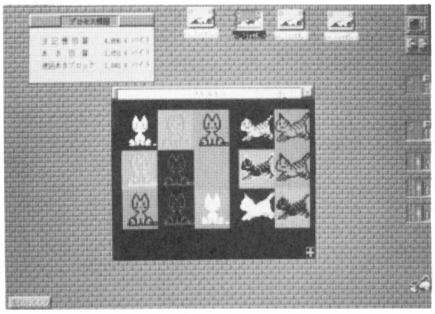


写真3　リスト6の実行例

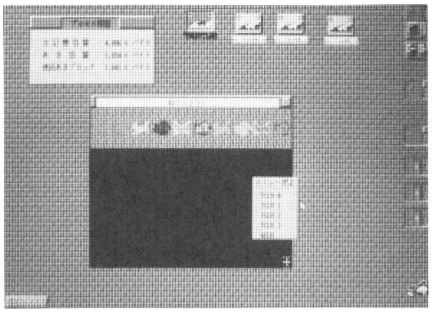


写真4　リスト7の実行例

リスト５の実行結果を写真２に示します。

●リスト６…GMCopy関数を用いて，グラフィックタイプのビットイメージをグラフィック画面に，テキストタイプのビットイメージをテキスト画面に，縦横２倍に拡大しながらコピーします。仮想的なビットマップを作り出して現在のグラフポートのビットマップにコピーするのはリスト５のプログラムと同様です。リスト６ではコピーモードを適当に変更しながら同じイメージを何度かコピーしています。

リスト６の実行結果を写真３に示します。

●リスト７…GMPlotImg関数の使用例です。２種類のプロットイメージを，交互に，プロットモードを順次変更しながら描画し，プロットモードの効果を確かめられるようにしてあります。描画時のプロットモードはウィンドウのタイトルに表示するようにしてあります。

リスト７の実行結果を写真４に示します。

リストの説明は以上ですが，コンパイルの方法をここにまとめておきます。リスト３のプログラムがmain.cというファイル名，リスト４～リスト７のプログラムのひとつがsubr.cというファイル名であるとします。このとき，実行型のファイルを作るためには，GCCを使用するなら，

gcc -oprog.x main.c subr.c -lsx -liocs

によってコンパイルすればよいでしょう。XCを使用するなら，

cc /Fxprog.x main.c subr.c/Ysxlib.a

によってコンパイルすることになります(sxlib.aがカレントドライブにある場合)。これによって，prog.xという実行ファイルができあがります。リスト３～リスト７のプログラムではiocslib.aの関数を使用していますから，そのためのスイッチ(-liocs, /Y）をかならずつけてください。

終わりに

今回はビットイメージを扱ってみました。ビットイメージはパソコンのディスプレイに描画する図形としてはもっとも基本的なものですが，基本的であるがゆえにそのデータ構造は画面モードの構成の影響をモロに受けています。ビットイメージの種類も画面モードを反映して複数用意されていますが，現時点ではテキストタイプのビットイメージとG16くらいしか使いものにならないというのが実感です。逆にいえば，この２種類のビットイメージさえ押さえておけばSX-WINDOWのイメージ処理は完璧なのかもしれません。

さて，何回かにわたってグラフマンの関数について説明してきましたが，グラフマンは今回で一応終わりにします。もちろん，説明をしていない重要な関数もあると思います。これらは必要に応じて説明することにしましょう。次回からは，タスクマンの

次回からは，タスクマンの関数に関して説明していきたいと思っています。それでは次回まで。

＜参考文献＞

1) 吉沢正敏，SX-WINDOWプログラミング，ソフトバンク，1991年.

2) 吉沢正敏，追補版SX-WINDOWプログラミング，ソフトバンク，1991年.

リスト１

```
/*
**********************************************

    テキストタイプのビットイメージの例
    ファイル名:sx_pat.c

**********************************************
*/
/*
          ねこのパターン  32×32ドット
*/
unsigned short  neko1[]={

 0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0111000000000000
,0b0000000000000001,0b1000110000000000
,0b0000000000000110,0b0111001100000000
,0b0000000000001001,0b1000100010000000
,0b0000000000011100,0b0110010010000000
,0b0011000000100011,0b0001001001000000
,0b0011100000111000,0b1000000000100000
,0b0011111001000110,0b0000000000110000
,0b0001000111000001,0b1000000001001000
,0b0001000100100001,0b0000000011010100
,0b0001000010010001,0b1111100010100110
,0b0010000001000010,0b0000000010011010
,0b0100100000000011,0b1110000001001001
,0b0100100000000010,0b0000000000100101
,0b0100100011100001,0b1110000001000011
,0b1100001110000001,0b0000000010000000
,0b1000000000000010,0b1100000010000000
,0b0100000000000011,0b0111100110000000
,0b0010000011000100,0b1000100110000000
,0b0001111101111001,0b0111001001000000
,0b0000000000101110,0b0101001101000000
,0b0000000000010111,0b1111010011000000
,0b0000000000001001,0b0001110000000000
,0b0000000000000111,0b0001100000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000

,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0111000000000000
,0b0000000000000001,0b1000110000000000
,0b0000000000000110,0b0111001100000000
,0b0000000000001001,0b1000100010000000
,0b0000000000011100,0b0110010010000000
,0b0011000000100011,0b0001001001000000
,0b0011100000111000,0b1000000000100000
,0b0011111001000110,0b0000000000110000
,0b0001000111000001,0b1000000001001000
,0b0001000100100001,0b0000000011010100
,0b0001000010010001,0b1111100010100110
,0b0010000001000010,0b0000000010011010
,0b0100100000000011,0b1110000001001001
,0b0100100000000010,0b0000000000100101
,0b0100100011100001,0b1110000001000011
,0b1100001110000001,0b0000000010000000
,0b1000000000000010,0b1100000010000000
,0b0100000000000011,0b0111100110000000
,0b0010000011000100,0b1000100110000000
,0b0001111101111001,0b0111001001000000
,0b0000000000101110,0b0101001101000000
,0b0000000000010111,0b1111010011000000
,0b0000000000001001,0b0001110000000000
,0b0000000000000111,0b0001100000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000

,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0111000000000000
,0b0000000000000001,0b1000110000000000
,0b0000000000000110,0b0111011100000000
,0b0000000000000011,0b1001101100000000
,0b0000000000011100,0b1110110110000000
,0b0000000000000111,0b0111111111000000
,0b0000000000111001,0b1111111111000000
,0b0000111000111110,0b0111111110110000
,0b0000111011011110,0b1111111100101000
,0b0000111101101110,0b0000011100011000
,0b0001111110111101,0b1111111100000100
,0b0011011111111100,0b0001111110000110
,0b0011011111111101,0b1111111111000010
,0b0011011100011110,0b0001111110000000
,0b0011110001111110,0b1111111100000000
,0b0111111111111101,0b0011111100000000
,0b0011111111111100,0b1000011000000000
,0b0001111100111011,0b0111011000000000
,0b0000000000000110,0b1000110110000000
,0b0000000000010001,0b1010110010000000
,0b0000000000001000,0b0000100000000000
,0b0000000000000110,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000

,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0111000000000000
,0b0000000000000001,0b1111100000000000
,0b0000000000000111,0b1111111000000000
,0b0000000000001111,0b1111111110000000
,0b0000000000011111,0b1111111110000000
,0b0011000000111111,0b1111111111000000
,0b0011100000111111,0b1111111111100000
,0b0011111001111111,0b1111111111110000
,0b0001111111111111,0b1111111111111000
,0b0001111111111111,0b1111111111111100
,0b0001111111111111,0b1111111110111110
,0b0011111111111111,0b1111111110011110
,0b0111111111111111,0b1111111111001111
,0b0111111111111111,0b1111111111100111
,0b0111111111111111,0b1111111111000011
,0b1111111111111111,0b1111111110000000
,0b1111111111111111,0b1111111110000000
,0b0111111111111111,0b1111111110000000
,0b0011111111111111,0b1111111110000000
,0b0001111101111111,0b1111111111000000
,0b0000000000111111,0b1111111111000000
,0b0000000000011111,0b1111110011000000
,0b0000000000001111,0b0001110000000000
,0b0000000000000111,0b0001100000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
};

unsigned short  neko2[]={

 0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000010000000000,0b0000000000000111
,0b0000011000000000,0b0000000000001111
,0b0000011100000000,0b0000000000010001
,0b0000011110000000,0b0000000000111010
,0b0000010011000000,0b0000000001000100
,0b0000110000110000,0b0000000011101000
,0b0001000000101000,0b0000001100010000
,0b0010010000101100,0b0001110010010000
,0b0010010001001100,0b0110011001010000
,0b0010010000001010,0b1011000100101000
,0b0100000000010011,0b0000100010001000
,0b0100001111000000,0b1100010001001000
,0b0100000000000010,0b0010001000001000
,0b0010000000000001,0b0001000000001000
,0b0001111000000000,0b0000000000000100
,0b0000000111100000,0b0001000001000100
,0b0000000100000000,0b0000110110001011
,0b0000000111000000,0b0000001110110001
,0b0000011001011000,0b0001111001111001
,0b0000100100101111,0b1110000111100101
,0b0001010011011100,0b0000000000010011
,0b0010001100100000,0b0000000000001100
,0b0001110011000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000

,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000010000000000,0b0000000000000111
,0b0000011000000000,0b0000000000001111
,0b0000011100000000,0b0000000000010001
,0b0000011110000000,0b0000000000111010
,0b0000010011000000,0b0000000001000100
,0b0000110000110000,0b0000000011101000
,0b0001000000101000,0b0000001100010000
,0b0010010000101100,0b0001110010010000
,0b0010010001001100,0b0110011001010000
,0b0010010000001010,0b1011000100101000
,0b0100000000010011,0b0000100010001000
,0b0100001111000000,0b1100010001001000
,0b0100000000000010,0b0010001000001000
,0b0010000000000001,0b0001000000001000
,0b0001111000000000,0b0000000000000100
,0b0000000111100000,0b0001000001000100
,0b0000000100000000,0b0000110110001011
,0b0000000111000000,0b0000001110110001
,0b0000011001011000,0b0001111001111001
,0b0000100100101111,0b1110000111100101
,0b0001010011011100,0b0000000000010011
,0b0010001100100000,0b0000000000001100
,0b0001110011000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000

,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
```

```
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000001110
,0b0000000000000000,0b0000000000000100
,0b0000000110000000,0b0000000000111000
,0b0000000111000000,0b0000000000010000
,0b0000111111010000,0b0000000011100000
,0b0001101111010000,0b0000000110110000
,0b0001101110110000,0b0001100110100000
,0b0001101111110100,0b0100111011010000
,0b0011111111101100,0b1111011101110000
,0b0011110000111111,0b0011101110110000
,0b0011111111111101,0b1101110111110000
,0b0001111111111110,0b1110111111110000
,0b0000000111111111,0b1111111111111000
,0b0000000000011111,0b1110111110111000
,0b0000000011111111,0b1111001001110100
,0b0000000001111111,0b1111110001001110
,0b0000000110100111,0b1110000110000110
,0b0000011011010000,0b0000000000011010
,0b0000010110010000,0b0000000000001100
,0b0001110011000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000

,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000010000000000,0b0000000000000111
,0b0000011000000000,0b0000000000001111
,0b0000011100000000,0b0000000000011111
,0b0000011110000000,0b0000000000111110
,0b0000011111000000,0b0000000001111100
,0b0000111111110000,0b0000000011111000
,0b0001111111111000,0b0000000111110000
,0b0011111111111100,0b0001111111110000
,0b0011111111111100,0b0111111111110000
,0b0011111111111110,0b1111111111111000
,0b0111111111111111,0b1111111111111000
,0b0111111111111111,0b1111111111111000
,0b0111111111111111,0b1111111111111000
,0b0011111111111111,0b1111111111111000
,0b0001111111111111,0b1111111111111100
,0b0000000111111111,0b1111111111111100
,0b0000000111111111,0b1111111111111111
,0b0000001111111111,0b1111111111111111
,0b0000011111111111,0b1111111111111111
,0b0000111111111111,0b1110000111111111
,0b0001111111111100,0b0000000000011111
,0b0011111111100000,0b0000000000001100
,0b0001110011000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
,0b0000000000000000,0b0000000000000000
};
```

リスト2

```
/*
*****************************************************

    グラフィックタイプのビットイメージの例
    ファイル名：sx_pat2.c

*****************************************************
*/
#define W   8   /* 白 */
#define B   11  /* 黒 */
#define G   14  /* 緑 */

/*
     グラフィックタイプのビットイメージ

     白色のねこ
*/

short  neko_gr[]={

 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,B,0,0,0,0,0,0,B,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,B,W,B,0,0,0,0,B,W,B,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,B,W,W,B,0,0,0,B,W,W,B,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,B,W,W,W,B,0,0,0,B,W,W,W,B,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,B,W,W,W,W,B,0,B,W,W,W,W,B,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,B,W,W,W,W,W,B,B,B,W,W,W,W,W,B,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,B,W,W,W,W,W,W,W,W,W,W,W,W,W,B,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,B,W,W,W,B,W,W,W,W,W,B,W,W,W,B,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,B,W,W,W,B,W,W,W,W,W,B,W,W,W,B,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,B,W,W,W,B,W,W,W,W,W,B,W,W,W,B,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,B,W,W,W,W,W,W,W,W,W,W,W,W,W,B,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,B,W,B,B,B,W,W,B,W,W,B,B,B,W,B,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,B,W,W,W,W,W,W,W,W,W,W,W,B,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,B,W,W,W,W,W,W,W,W,W,B,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,B,B,B,W,W,W,B,B,B,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,B,W,W,W,B,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,B,W,W,W,W,W,B,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,B,W,W,W,W,W,W,W,B,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,B,W,W,W,W,W,W,W,W,W,B,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,B,B,0,B,W,W,W,W,W,W,W,W,W,B,0,B,B,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,B,W,W,B,B,W,W,W,W,B,W,W,W,W,B,B,W,W,B,0,0,0,0,0,0
,0,0,0,0,0,0,B,W,W,W,W,B,B,W,W,W,B,W,W,W,B,B,W,W,W,W,B,0,0,0,0,0
,0,0,0,0,0,0,B,W,W,W,W,W,B,W,W,W,B,W,W,W,B,W,W,W,W,W,B,0,0,0,0,0
,0,0,0,0,0,0,B,B,B,W,W,W,B,W,W,W,B,W,W,W,B,W,W,W,B,B,B,B,B,B,B,0
,0,0,0,0,0,B,B,W,W,W,W,W,B,W,W,B,B,B,W,W,B,W,W,W,W,W,B,B,W,W,W,B
,0,0,0,0,0,B,B,B,B,B,B,B,B,B,B,0,B,B,B,B,B,B,B,B,B,B,B,B,B,B,B,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

};

/*
     GR2（G16）タイプのビットイメージ

     明るいグレーのねこ
*/

char    neko_gr2[]={

 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0xbb,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0xb9,0xbb,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0xb9,0x99,0xb0,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0xb9,0x99,0x9b,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x0b,0xbb,0x00,0x00,0xb9,0x99,0x99,0xbb,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x0b,0x99,0xbb,0xb0,0xbb,0x99,0x99,0x99,0xb0,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x0b,0x99,0x99,0x9b,0xb9,0x99,0x99,0x99,0x9b,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0xb9,0x99,0x99,0x99,0x99,0xb9,0x99,0x99,0xb0,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0xb9,0x99,0x99,0x99,0x99,0x9b,0x99,0x99,0xb0,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0xb9,0x99,0x99,0x99,0x99,0x99,0x99,0xb9,0xb0,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0xb9,0x99,0x9b,0x99,0x99,0x99,0x9b,0x99,0xb0,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x0b,0x99,0x99,0xb9,0x99,0xb9,0x99,0x99,0xb0,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x0b,0x99,0x99,0x99,0x99,0x99,0xbb,0xbb,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0xb9,0x9b,0xbb,0x99,0x9b,0x99,0x9b,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x0b,0x99,0x99,0x99,0x99,0xb9,0x99,0xb0,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0xbb,0xbb,0xb9,0x99,0xbb,0xbb,0xb0,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xb9,0x99,0xb9,0x99,0xb0,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x0b,0x99,0x9b,0xb9,0x99,0xb0,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0xb9,0x99,0x99,0x99,0x99,0xb0,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x0b,0x99,0x99,0x99,0x99,0xbb,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0xbb,0x0b,0x99,0x99,0x99,0xb9,0x9b,0x0b,0xb0,0x00,0x00,0x00
,0x00,0x00,0x00,0x0b,0x99,0xbb,0x99,0x99,0xbb,0x99,0x9b,0xb9,0x9b,0x00,0x00,0x00
,0x00,0x00,0x00,0xb9,0x99,0x9b,0xb9,0x99,0xb9,0x99,0x99,0x99,0x99,0xb0,0x00,0x00
,0x00,0x00,0x00,0xb9,0x99,0x99,0xb9,0x99,0xb9,0x99,0x99,0x99,0x99,0xbb,0xbb,0x00
,0x00,0x00,0x00,0xbb,0xb9,0x99,0xb9,0x99,0xb9,0xb9,0x99,0x99,0xbb,0xb9,0x99,0xb0
,0x00,0x00,0x0b,0xb9,0x99,0x99,0xb9,0x9b,0xbb,0x99,0x99,0x99,0x99,0xbb,0xbb,0x00
,0x00,0x00,0x0b,0xbb,0xbb,0xbb,0xbb,0xbb,0x0b,0xbb,0xbb,0xbb,0xbb,0xbb,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00

};

/*
     GR3タイプのビットイメージ

     緑色のねこ
*/

char    neko_gr3[]={

 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,B,B,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,B,G,B,B,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,B,G,G,G,B,B,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,B,G,G,G,G,G,B,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,B,G,G,G,G,G,G,B,B,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,B,B,B,B,0,0,0,B,G,G,G,G,G,G,G,B,0,B,B,0,0,0,0,0,0,0,0,0,0,0
,0,0,B,G,G,G,B,B,B,B,G,G,G,G,G,G,G,G,B,B,G,B,0,0,0,0,0,0,0,0,0,0
,0,0,0,B,G,G,G,G,G,G,G,G,G,G,G,G,G,G,B,B,G,B,0,0,0,0,0,0,0,0,0,0
,0,0,0,B,G,G,G,G,G,G,G,G,B,G,G,G,G,G,B,B,G,B,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,B,G,G,G,G,G,G,G,B,G,G,G,G,G,B,B,G,B,B,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,B,G,G,G,G,G,B,G,G,G,G,G,G,B,B,G,G,G,B,0,0,0,0,0,0,0,0
,0,0,0,0,0,B,G,G,G,G,G,B,G,G,B,B,G,G,G,B,B,G,G,G,G,B,0,0,0,0,0,0
,0,0,0,0,0,B,B,B,B,G,G,G,G,B,G,G,G,G,G,B,B,G,G,G,G,G,B,0,0,0,0,0
,0,0,0,0,0,B,G,G,G,G,G,G,G,G,G,G,G,B,0,B,B,G,G,G,G,G,B,0,0,0,0,0
,0,0,0,0,0,B,B,B,B,G,B,G,G,G,G,G,G,B,0,0,B,G,G,G,G,G,B,0,0,0,0,0
,0,0,0,0,0,0,B,B,G,G,G,G,G,B,B,G,G,G,B,B,B,G,G,G,G,G,B,0,0,0,0,0
,0,0,0,0,0,0,0,0,B,B,B,B,B,G,G,G,G,G,G,G,B,G,G,G,G,G,B,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,B,G,G,G,G,G,G,G,G,B,G,G,G,G,G,B,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,B,B,0,B,G,G,G,G,G,G,G,G,B,G,G,G,G,G,B,0,0,0,0,0
,0,0,0,0,0,0,0,B,G,G,B,B,G,G,G,G,B,G,G,G,G,B,G,G,G,G,B,B,0,0,0,0
,0,0,0,0,0,0,B,G,G,G,G,B,B,G,G,G,B,G,G,G,B,B,G,G,G,G,B,0,0,0,0,0
,0,0,0,0,0,0,B,G,G,G,G,G,B,G,G,G,B,G,G,G,B,G,G,G,G,B,0,0,0,0,0,0
,0,0,0,0,0,0,B,B,B,G,G,G,B,G,G,G,B,G,G,G,B,G,G,B,B,B,B,B,B,B,B,0
,0,0,0,0,0,B,B,G,G,G,G,G,B,G,G,B,B,B,G,G,B,B,B,B,G,G,G,G,G,G,G,B
,0,0,0,0,0,B,B,B,B,B,B,B,B,B,B,0,B,B,B,B,B,B,B,B,B,B,B,B,B,B,B,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

};
```

リスト3

```
/*

     SX-WINDOWスケルトンプログラム Ver.2.2

              ダウン・サイジング版

                         (C) 中森 章, Apr.18, 1992

     DRAW() と IDLE() を外部で定義することによって
     ウィンドウ上に絵を描くことができる

*/
#include <stdio.h>
#define __POINT_T    /* point_t 型を使う */
#include <sxlib.h>
#define FALSE    0
#define TRUE     ~FALSE

#define MDEFID       1
#define MNILIST      "TCLR 0,TCLR 1,TCLR 2,TCLR 3,GCLR"
#define MNTITLE           "¥014メニューだよ"
menu     **menuHdl;

/*
     ここでウィンドウに関する定数を設定
*/
#define WDEFID       WI_STDG2
#define WINOPT       ( WC_GBOX | WC_GBOXON )
#define WINWIDTH     360
#define WINHIGHT     256
#define WINTITLE     "¥012スケルトン"
#define EVENTMASK    EM_EVERY

#define MAXWIDTH     700
#define MAXHIGHT     700
#define MINWIDTH     100
#define MINHIGHT     18
/*
     ここは定数から計算される定数
*/
```

▶「(影)のショートプロぱーてぃ」で挿絵はキツネになっているのに文中の絵は牛のままだ。
加藤　安弘(17)滋賀県

```
#define WINOPTL      ( WINOPT & 0xf )
#define WINDEFID     ( WDEFID << 4 | WINOPTL )

window  *winPtr;
rect    winSize;
rect    winMinMax={MINWIDTH,MINHIGHT,MAXWIDTH,MAXHIGHT};
event   eventRec;
int activeFlag;
int ctrlFlag;     /* コントロールがあるかないか */
int menuFlag;     /* メニューがあるかないか */

#ifdef __GNUC__
asm( ".xdef      _STACK_SIZE"  );
asm( "_STACK_SIZE equ   8192"  );
asm( ".xdef      _HEAP_SIZE"   );
asm( "_HEAP_SIZE equ    16384" );
#endif

main()
{
    if( SX_init()==FALSE ){
        DMError(0x101,"ウィンドウがオープンできません");
        exit();
    }
    while( 1 ){
        TSEventAvail(EVENTMASK,(tsevent*)&eventRec);
        switch( eventRec.eWhat ){
        case E_IDLE:      procIDLE();     break;
        case E_MSLDOWN:   procMSLDOWN(); break;
        case E_MSLUP:     procMSLUP();   break;
        case E_MSRDOWN:   procMSRDOWN(); break;
        case E_MSRUP:     procMSRUP();   break;
        case E_KEYDOWN:   procKEYDOWN(); break;
        case E_KEYUP:     procKEYUP();   break;
        case E_UPDATE:    procUPDATE();  break;
        case E_ACTIVATE:  procACTIVATE();break;
        case E_SYSTEM1:   procSYSTEM();  break;
        case E_SYSTEM2:   procSYSTEM();  break;
        case E_USER1:     procUSER();    break;
        case E_USER2:     procUSER();    break;
        }
    }
}

SX_init()
{
    task    taskBuf;
    char    BUF[100];

    TSGetTdb(&taskBuf, -1);
    if( (TSTakeParam(&taskBuf.command,&winSize,NULL,0,NULL,NULL)&1)==0 ){
        *(int *)&winSize.left = TSGetWindowPos();
        winSize.right  = winSize.left + WINWIDTH;
        winSize.bottom = winSize.top  + WINHIGHT;
    }
    winPtr=WMOpen(NULL,&winSize,(LASCII*)WINTITLE,TRUE,WINDEFID,(window *)-1,TRUE,TSGetID());
    if( winPtr == NULL ) return( FALSE );
    winPtr->wOption = WINOPT;
    activeFlag=FALSE;
    ctrlFlag  = CtrlPrepare();/* コントロールが不要なら ctrlFlag=FALSE */
    menuFlag  = MenuPrepare();/* メニューが不要なら     menuFlag=FALSE */
    DRAW();
    drawGrowBox();
    return( TRUE );
}

SX_term()
{
    if( ctrlFlag ) CtrlDispose();
    WMDispose( winPtr );
    exit();
}

drawGrowBox()
{
    GMSetGraph( &winPtr->wGraph );
    WMDrawGBox( winPtr );
}

CtrlPrepare()
{
    return( FALSE );
}

CtrlDispose()
{
    return( FALSE );
}

MenuPrepare()
{
    menuHdl=MNConvert(0,MNILIST,MDEFID);
    if(menuHdl<(menu**)0) return( FALSE );
#if MDEFID==1
    (*menuHdl)->mHandle=(long)MNTITLE;
#endif
    return( TRUE );
}

MenuDispose()
{
    MMHdlDispose(menuHdl);
    return( TRUE );
}

procIDLE()
{
    IDLE();
}

procMSLDOWN()
{
    if( (window*)eventRec.eWhom != winPtr ) return( FALSE );
    if( activeFlag == FALSE ){
        WMSelect( winPtr );
        activeFlag = TRUE;
        if( EMLStill() == 0) goto checkDClick;
    }
    switch( SXCallWindM2(winPtr,(tsevent*)&eventRec,&winMinMax) ){
    case W_INCLOSE:
        SX_term(); break;
    case W_INGROW:
    case W_INZMOUT:
    case W_INZMIN:
        GMClipRect(&winPtr->wGraph.grRect);
        break;
    }
checkDClick:
    TSGetEvent(EVENTMASK,(tsevent*)&eventRec);
    return( TRUE );
}

procMSLUP()
{
    return( FALSE );
}

procMSRDOWN()
{
    int item;

    if( (window*)eventRec.eWhom != winPtr ) return( FALSE );
    GMSetGraph( &winPtr->wGraph );
    if( activeFlag == FALSE ) return( FALSE );
    item=MNSelect(menuHdl,eventRec.eWhere);
    TSGetEvent(EVENTMASK,(tsevent*)&eventRec);
    switch(item){
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
        TCLR(item-1); break;
    }
    return( TRUE );
}

procMSRUP()
{
    return( FALSE );
}

procKEYDOWN()
{
    return( FALSE );
}

procKEYUP()
{
    return( FALSE );
}

procUPDATE()
{
    if( (window*)eventRec.eWhom != winPtr ) return( FALSE );
    WMUpdate( winPtr );
    if( ctrlFlag ) CMDraw( winPtr );
    DRAW();
    WMUpdtOver( winPtr );
    drawGrowBox();
    TSGetEvent(EVENTMASK,(tsevent*)&eventRec);
}

procACTIVATE()
{
    if( (window*)eventRec.eWhom == winPtr )    activeFlag = TRUE;
    else if( eventRec.eWhom != NULL ){
        if( activeFlag ) {
            activeFlag = FALSE;
            TSGetEvent(EVENTMASK,(tsevent*)&eventRec);
        }
    }
    return( TRUE );
}

procSYSTEM()
{
    switch( ((tsevent*)&eventRec)->what2 ){
    case CLOSEALL:
    case ENDTSK:
        SX_term(); break;
    case WINDOWSELECT:
        WMSelect( winPtr ); break;
    }
}

procUSER()
{
    return( FALSE );
}

TCLR(n)
int  n;
{
    int addr;
    int i;

    DMWaitOpen();
```

▶「Inside X68000」が6,800円，高いなあ。と思って数週間後見てみると，いつのまにか，第3刷！　早く買っておけばよかった。第1刷がほしかったよ。

井上　敬介(21)神奈川県

```
    if(n!=4){
        addr=0xe00000+0x020000*n;
        for(i=0;i<0x20000;i+=4){
            B_LPOKE(addr+i,0);
            if((i&0xfff)==0) DMWaitWhile();
        }
    }
    else {
        addr=0xc00000;
        for(i=0;i<0x200000;i+=4){
            B_LPOKE(addr+i,0);
            if((i&0xfff)==0) DMWaitWhile();
        }
    }
    DMWaitClose();
}
```

リスト4

```
/*
********************************************************

    GMPutRImg GMPutImg GMDrawG16 を用いた描画

********************************************************
*/
#include <stdio.h>
#define __POINT_T   /* point_t 型を使う */
#include <sxlib.h>
#include "sx_pat.c" /* テキストタイプのビットイメージ */
#include "sx_pat2.c"/* グラフィックタイプのビットイメージ */

extern window* winPtr;

short temp[sizeof(neko_gr)]={ /* 適当な大きさの作業用配列 */
0,0,32,32
};

/*
    GMGetBitmap にはバグがあるので....
*/
#ifdef __GNUC__
asm(".text");
asm("_GMGetBitmap:");
asm(" .dc.w $a1c7");
asm(" move.l a0,d0");
asm(" rts");
#else
#asm
    .text
_GMGetBitmap:
    .dc.w   $a1c7
    move.l  a0,d0
    rts
#endasm
#endif
/*
    ヌルイベント時の処理
*/
IDLE()
{}

/*
    テキストのパレットをグラフィックのパレットにコピー
*/
setPalet()
{
    int i;

    for(i=0;i<16;i++)
        B_WPOKE(0xe82000+i*2,B_WPEEK(0xe82200+i*2));
}

/*
    いろいろなイメージを描画する
*/
DRAW()
{
    point_t p;
    bitmap  *bm0,bm;
    rect    r;
    int i;

    GMSetGraph( &winPtr->wGraph );

    GMAPage(15);
    bm0=GMGetBitmap();        /* 元のビットマップ(テキストタイプ) */

    bm.bmKind       =G_GRP; /* グラフィックタイプのビットマップを作る */
    bm.bmRect       =bm0->bmRect;
    bm.base         =0xc00000;
    bm.line         =2048;
    bm.opt.bRatio   =0x8000;

    setPalet();                    /* パレットを設定 */

    GMSetBitmap( &bm );       /* グラフィックタイプのビットマップに変更 */

    /* ここからはグラフィック画面に描画される */

    /* SX本などに GMPutRImg はグラフィックタイプのビットマップに */
    /* 描画できるような記述があるが, GMPutRImg はかならず */
    /* テキストタイプのビットマップに描画する */

    memcpy(temp+4,neko_gr,sizeof(neko_gr));  /* rect image を作る */
    p.x_y=0x00100010;
    GMPutRImg((rectImg*)temp,p);                  /* 描画されない！ */

    r.left  =0x10;
    r.top   =0x40;
    r.right =0x30;
    r.bottom=0x60;
    GMPutImg((unsigned short*)neko_gr,&r);     /* 描画されない！ */

    memcpy(temp+4,neko_gr2,sizeof(neko_gr2)); /* rect image を作る */
    p.x_y=0x00100070;
    GMDrawG16((tx16*)temp,p);

    GMExgBitmap( bm0 );    /* テキストタイプのビットマップと交換 */

    /* ここからはテキスト画面に描画される */

    memcpy(temp+4,neko1,sizeof(neko1));  /* rect image を作る */
    p.x_y=0x00900010;
    GMPutRImg((rectImg*)temp,p);

    r.left  =0x90;
    r.top   =0x40;
    r.right =0xb0;
    r.bottom=0x60;
    GMPutImg((unsigned short*)neko2,&r);

    memcpy(temp+4,neko_gr2,sizeof(neko_gr2)); /* rect image を作る */
    p.x_y=0x00900070;
    GMDrawG16((tx16*)temp,p);

}
```

リスト5

```
/*
**********************************

    GMTransTmg を用いた描画

**********************************
*/
#include <stdio.h>
#define __POINT_T   /* point_t 型を使う */
#include <sxlib.h>
#include "sx_pat.c" /* テキストタイプのビットイメージ */
#include "sx_pat2.c"/* グラフィックタイプのビットイメージ */

extern window* winPtr;

/*
    ヌルイベント時の処理
*/
IDLE()
{}


/*
    いろいろなタイプのイメージデータを
    現在のグラフポートに描画する関数
*/
PutImg(kind,img,pt)
int kind;
short   *img;
point_t pt;
{
    bitmap  bm;
    rect    srect,drect;

    bm.bmKind       =kind;
    bm.bmRect.left  =0;           /* 32×32ドット固定 */
    bm.bmRect.top   =0;
    bm.bmRect.right =32;
    bm.bmRect.bottom=32;
    bm.base         =(int)img;
    bm.opt.bRatio=0x8000;

    switch(kind){
    case G_TXT:
        bm.line=4;                 /* 横のバイト数 */
        bm.opt.tbm.page=4*32;      /* 1ページ分のバイト数 */
        bm.opt.tbm.aPage=15;       /* アクセスページ */
        break;
    case G_GRP:
        bm.line=64; break;
    case G_GR2:
        bm.line=16; break;
    case G_GR3:
        bm.line=32; break;
    }
    srect       =bm.bmRect;
    drect.left  =pt.p.x;
    drect.top   =pt.p.y;
    drect.right =pt.p.x+32;
    drect.bottom=pt.p.y+32;

    GMSetGraph(&winPtr->wGraph);
```

▶かまたさんへ。「キーボード操作のウィンドウシステムってあるんだろうか？」とのことですが，お答えしましょう。ずばり「SYSTEM-7C」があります！

野田　敏之(20)神奈川県

```c
    GMTransImg(&bm,winPtr->wGraph.bmap,&srect,&drect);
}

/*
    GMGetBitmap にはバグがあるので....
*/
#ifdef __GNUC__
asm(".text");
asm("_GMGetBitmap:");
asm(" .dc.w $a1c7");
asm(" move.l a0,d0");
asm(" rts");
#else
#asm
    .text
_GMGetBitmap:
    .dc.w   $a1c7
    move.l  a0,d0
    rts
#endasm
#endif

/*
    テキストのパレットをグラフィックのパレットにコピー
*/
setPalet()
{
    int i;

    for(i=0;i<16;i++)
        B_WPOKE(0xe82000+i*2,B_WPEEK(0xe82200+i*2));
}

/*
    いろいろなイメージを描画する
*/
DRAW()
{
    point_t p;
    bitmap  *bm0,bm;
    int i;

    GMSetGraph( &winPtr->wGraph );

    GMAPage(15);
    bm0=GMGetBitmap();          /* 元のビットマップ（テキストタイプ） */

    bm.bmKind       =G_GRP; /* グラフィックタイプのビットマップを作る */
    bm.bmRect       =bm0->bmRect;
    bm.base         =0xc00000;
    bm.line         =2048;
    bm.opt.bRatio   =0x8000;

    setPalet();                 /* パレットを設定 */

    GMSetBitmap( &bm );         /* グラフィックタイプのビットマップに変更 */

    /* ここからはグラフィック画面に描画される */

    p.x_y=0x00100010;
    PutImg(G_GRP,(rectImg*)neko_gr,p);
    p.x_y=0x00100040;
    PutImg(G_GR2,(rectImg*)neko_gr2,p);
    p.x_y=0x00100070;
    PutImg(G_GR3,(rectImg*)neko_gr3,p);
    p.x_y=0x001000a0;
    PutImg(G_TXT,(rectImg*)neko2,p);

    GMExgBitmap( bm0 );         /* テキストタイプのビットマップと交換 */

    /* ここからはテキスト画面に描画される */

    p.x_y=0x00900010;
    PutImg(G_GRP,(rectImg*)neko_gr,p);
    p.x_y=0x00900040;
    PutImg(G_GR2,(rectImg*)neko_gr2,p);
    p.x_y=0x00900070;
    PutImg(G_GR3,(rectImg*)neko_gr3,p);
    p.x_y=0x009000a0;
    PutImg(G_TXT,(rectImg*)neko2,p);    /* GMPutRImg(txtImg,p)と同じ */
}
```

リスト6

```c
/*
*******************************************

    GMCopy を用いた描画（拡大つき）

*******************************************
*/
#include <stdio.h>
#define __POINT_T    /* point_t 型を使う */
#include <sxlib.h>
#include "sx_pat.c" /* テキストタイプのビットイメージ */
#include "sx_pat2.c"/* グラフィックタイプのビットイメージ */

extern window* winPtr;

/*
    ヌルイベント時の処理
*/
IDLE()
{}

/*
    いろいろなタイプのイメージデータを
    縦、横それぞれ2倍に拡大して
    現在のグラフポートに描画する関数
*/
CpyImg(kind,img,pt,mode)
int kind;
short   *img;
point_t pt;
int     mode;
{
    bitmap  bm;
    rect    srect,drect;

    bm.bmKind       =kind;
    bm.bmRect.left  =0;             /* 32×32ドット固定 */
    bm.bmRect.top   =0;
    bm.bmRect.right =32;
    bm.bmRect.bottom=32;
    bm.base         =(int)img;
    bm.opt.bRatio=0x8000;

    switch(kind){
    case G_TXT:
        bm.line=4;                  /* 横のバイト数 */
        bm.opt.tbm.page=4*32;       /* 1ページ分のバイト数 */
        bm.opt.tbm.aPage=15;        /* アクセスページ */
        break;
    case G_GRP:
        bm.line=64; break;
    case G_GR2:
        bm.line=16; break;
    case G_GR3:
        bm.line=32; break;
    }
    srect       =bm.bmRect;/* srect は 32×32 ドット */
    drect.left  =pt.p.x;    /* drect を 64×64 ドットにする */
    drect.top   =pt.p.y;
    drect.right =pt.p.x+64;
    drect.bottom=pt.p.y+64;

    GMSetGraph(&winPtr->wGraph);
    GMCopy(&bm,winPtr->wGraph.bmap,&srect,&drect,mode,0);
}

/*
    GMGetBitmap にはバグがあるので....
*/
#ifdef __GNUC__
asm(".text");
asm("_GMGetBitmap:");
asm(" .dc.w $a1c7");
asm(" move.l a0,d0");
asm(" rts");
#else
#asm
    .text
_GMGetBitmap:
    .dc.w   $a1c7
    move.l  a0,d0
    rts
#endasm
#endif

/*
    テキストのパレットをグラフィックのパレットにコピー
*/
setPalet()
{
    int i;

    for(i=0;i<16;i++)
        B_WPOKE(0xe82000+i*2,B_WPEEK(0xe82200+i*2));
}

/*
    いろいろなイメージを描画する

    GMCopy は同じタイプのビットマップ間のコピーしかできない
*/
DRAW()
{
    point_t p;
    bitmap  *bm0,bm;
    int i;
    rect rg={0x10,0x10,0xd0,0xd0};
    rect rt={0xe0,0x10,0x160,0xd0};

    GMSetGraph( &winPtr->wGraph );

    GMAPage(15);
    bm0=GMGetBitmap();          /* 元のビットマップ（テキストタイプ） */

    bm.bmKind       =G_GRP; /* グラフィックタイプのビットマップを作る */
    bm.bmRect       =bm0->bmRect;
    bm.base         =0xc00000;
    bm.line         =2048;
    bm.opt.bRatio   =0x8000;

    setPalet();                 /* パレットを設定 */

    GMSetBitmap( &bm );         /* グラフィックタイプのビットマップに変更 */
    GMPenMode(G_PSET);
    GMForeColor(G_DGRAY);
    GMBackColor(G_THRU);

    /* ここからはグラフィック画面に描画される */

    GMFillRect( &rg ); /* 背景を塗る */
```

```
    p.x_y=0x00100010;
    CpyImg(G_GRP,(rectImg*)neko_gr,p,G_PSET);   /* pset */
    p.x_y=0x00100050;
    CpyImg(G_GRP,(rectImg*)neko_gr,p,G_NPSET); /* not pset */
    p.x_y=0x00100090;
    CpyImg(G_GRP,(rectImg*)neko_gr,p,G_OR);     /* or */
    p.x_y=0x00500010;
    CpyImg(G_GRP,(rectImg*)neko_gr,p,G_NOR);    /* nor */
    p.x_y=0x00500050;
    CpyImg(G_GRP,(rectImg*)neko_gr,p,G_ADD);    /* add */
    p.x_y=0x00500090;
    CpyImg(G_GRP,(rectImg*)neko_gr,p,G_SUB);    /* sub */
    p.x_y=0x00900010;
    CpyImg(G_GRP,(rectImg*)neko_gr,p,G_SELMAX); /* select max
*/
    p.x_y=0x00900050;
    CpyImg(G_GRP,(rectImg*)neko_gr,p,G_SELMIN); /* select min
*/
    p.x_y=0x00900090;
    CpyImg(G_GRP,(rectImg*)neko_gr,p,G_BLEND); /* blend */

    GMExgBitmap( bm0 );    /* テキストタイプのビットマップと交換 */
    GMPenMode(G_PSET);
    GMForeColor(G_DGRAY);
    GMBackColor(G_THRU);

    /* ここからはテキスト画面に描画される */

    GMFillRect( &rt ); /* 背景を塗る */

    p.x_y=0x00e00010;
    CpyImg(G_TXT,(rectImg*)neko2,p,G_PSET);   /* pset */
    p.x_y=0x00e00050;
    CpyImg(G_TXT,(rectImg*)neko2,p,G_NPSET); /* not pset */
    p.x_y=0x00e00090;
    CpyImg(G_TXT,(rectImg*)neko2,p,G_AND);    /* and */
    p.x_y=0x01200010;
    CpyImg(G_TXT,(rectImg*)neko2,p,G_OR);     /* or */
    p.x_y=0x01200050;
    CpyImg(G_TXT,(rectImg*)neko2,p,G_NAND);   /* nand */
    p.x_y=0x01200090;
    CpyImg(G_TXT,(rectImg*)neko2,p,G_NOR);    /* nor */
}
```

リスト7

```
/*
*********************************

      GMPlotImg を用いた描画

*********************************
*/
#include <stdio.h>
#define __POINT_T   /* point_t 型を使う */
#include <sxlib.h>
#include "sx_pat.c" /* テキストタイプのビットイメージ */
#include "sx_pat2.c"/* グラフィックタイプのビットイメージ */

extern window* winPtr;

short back[]={

0,0,16,16

,0b0010000010010000
,0b0001000000100000
,0b0001000001000000
,0b0001000110000000
,0b0000001000100000
,0b0000110000010000
,0b0011000000001100
,0b1100010000000011
,0b0000010000000000
,0b0000010000000000
,0b0000001000000000
,0b0000001000000000
,0b0000001000000000
,0b0000000100000000
,0b1100000100000011
,0b0010000100001100

,0b1011111011011111
,0b1101111110111111
,0b1101111001111111
,0b1101110111111111
,0b1111001110111111
,0b1100111111011111
,0b0011111111101110
,0b1111011111111111
,0b1111011111111111
,0b1111011111111111
,0b1111101111111111
,0b1111101111111111
,0b1111101111111111
,0b0111110111111100
,0b1111110111110011
,0b1011110111101111

,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0

,-1,-1,-1,-1,-1,-1,-1,-1
,-1,-1,-1,-1,-1,-1,-1,-1

};

/*
    ウィンドウに背景を描く
*/
WIPE()
{
        point_t p;
        int     x0,y0,x1,y1;
        int     x,y;

        GMSetGraph( &winPtr->wGraph );
        GMAPage(15);

        x0=winPtr->wGraph.grRect.left;
        y0=winPtr->wGraph.grRect.top;
        x1=winPtr->wGraph.grRect.right;
        y1=winPtr->wGraph.grRect.bottom;

        for(x=x0;x<=x1;x+=16){
                p.p.x=x;
                for(y=y0;y<=48;y+=16){
                        p.p.y=y;
                        GMPutRImg((rectImg*)back,p);
                }
        }
}

/*
        ２０回呼ばれるごとに描画
 */
IDLE()
{
        static  int     i=0;
        if(i<=0) {
                DRAW();
                i=20;
        }
        else {
                i--;
        }
}

/*
        ねこをパターンを変更しながら
        背景に重ね合わせて描く
*/

#define CHANGE

DRAW()
{
        static  rect    pos={100,16,132,48};
        static  int     pttn=1;
        static  int     protmd=0x300;

        GMSetGraph( &winPtr->wGraph );
        GMAPage(15);
        GMForeColor(G_RED);
        GMBackColor(G_GREEN);

#ifdef  CHANGE
        switch(protmd){ /* ウィンドウのタイトルを変更する */
        case 0x300: WMTitleSet(winPtr,(LASCII*)"¥014ねこ（標準）")
;     break;
        case 0x301: WMTitleSet(winPtr,(LASCII*)"¥014ねこ（反転）")
;     break;
        case 0x302: WMTitleSet(winPtr,(LASCII*)"¥015ねこ（ﾊｲﾗｲﾄ
）");      break;
        case 0x303: WMTitleSet(winPtr,(LASCII*)"¥021ねこ（ﾊｲﾗｲﾄ
反転）");  break;
        case 0x304: WMTitleSet(winPtr,(LASCII*)"¥014ねこ（消去）")
;     break;
        case 0x305: WMTitleSet(winPtr,(LASCII*)"¥014ねこ（？？）")
;     break;
        case 0x306: WMTitleSet(winPtr,(LASCII*)"¥016ねこ（網掛け）
");   break;
        case 0x307: WMTitleSet(winPtr,(LASCII*)"¥022ねこ（網掛け反
転）"); break;
        case 0x308: WMTitleSet(winPtr,(LASCII*)"¥016ねこ（不可視）
");   break;
        case 0x309: WMTitleSet(winPtr,(LASCII*)"¥022ねこ（不可視反
転）"); break;
        }
#endif
        if(pttn==1){
                pos.left   -= 32;
                pos.right  -= 32;
                GMPlotImg(neko1,&pos,protmd);
                pttn=2;
        }
        else{
                pos.left   -= 40;
                pos.right  -= 40;
                GMPlotImg(neko2,&pos,protmd);
                pttn=1;
        }
#ifdef  CHANGE
        protmd++;
        if(protmd==0x30a) protmd=0x300;
#endif
        if(pos.right<0){
                pos.left=winPtr->wGraph.grRect.right;
                pos.right=pos.left+32;
                WIPE(); /* 背景を描く */
        }
}
```

［特集］

プログラミング再入門

幸いなことに，Oh!Xでは「プログラミング」を勧める際に声を大にする必要はない。一般常識を遙かに超えるプログラミングユーザーを抱えてはいるが，ある程度の大きなプログラムを作ることのできる人はそう多いわけではない。これは技術力の問題というよりも，精神力，そして大規模プログラム作成のノウハウの問題であろう。
大きなプログラムは労力も大きい。これはある意味で当然のことだ。
それに負けない精神力を持つことがもっとも望ましいが，それは簡単に鍛えられるものではない。考えるべきは，いかに少ない労力で問題を解決するかである。
これは単純な技術ではない。むしろ考え方，に近いものがある。
まず，構想を練り，テンションを持続できる大きさのプログラムを作る。そして拡張する。これを繰り返すだけだ。この手法で目的まで到達可能だということ，を，知ること。これが大事だ。
そして，過程自体を楽しみ，結果に陶酔する。
考え方の問題である。
単純な手法の強力さは，「自分には無理だ」という思い込みがないかぎり，必ず成功に導かれるというところにある。
「自分の本当に作りたいプログラムを知る」こと。それ以前に「作りたいと思う」こと。これができたときにはすでに完成は目前にある。

CONTENTS

プログラミング言語の前に……中森　章　82
プログラムの流れをつかもう……文月　涼　84
正しい花瓶の落とし方……柴田　淳　90
作り散らかせます……丹　明彦　98
ちょっと大きいモノを書こう……横内威至　105

まず本質を知る
プログラミング言語の前に

Programing

Nakamori Akira 中森　章

たとえばC言語を始めようとして，言語自体の難解さでつまずく人もいます。しかし，本当に必要なのは頭の中でデータ構造と基本となるアルゴリズムを作ること。これがプログラミングなのです。

はじめに

私は編集部から依頼されて特集記事などに載せるプログラムを書くことがあります。いつも気楽に引き受けては締め切りとの格闘になるわけですが，このときプログラムが（技術的に）書けないと感じたことは一度もありません。でも，よく考えるとこれは不思議なことです。どうして，プログラムが書けるのでしょう？　今回の特集はプログラミング再入門ということですので，「プログラムを書く」とはどういうことなのかについて少し考えてみたいと思います。

プログラミング言語というもの

プログラムはプログラミング言語によって記述されます。コンピュータになにか仕事をさせたいときは，その手順をプログラミング言語で書く必要があります。そして，通常，手順をプログラミング言語に直す作業（プログラムを書くこと）がプログラミングと呼ばれています。ところが，真のプログラミングとはプログラムを書く時点ですでに終了しているのです。

ここで外国語について考えてみましょう。外国語は日本語に翻訳することにより，私たちが理解できるようになります。多くの場合，外国語の単語を1対1に別の外国語に置き換えることが可能です。これは，人間のあいだで共通の考えとか意思といったものが存在していて，言葉はそれを表現する道具にすぎないからなのです。当然，共通の意思の存在しない動物の言葉を人間の言葉に変換することは不可能だと思われます。

プログラミング言語も外国語と同じようなものです。プログラミング言語で記述されたプログラムはコンピュータに行わせたい処理をコンピュータにわかるように翻訳したものです。

我々の意志（目的）は処理そのものにあります。その処理をコンピュータに教えるとき，プログラミング言語はなんでも構いません。どのようなプログラミング言語でもコンピュータは理解できるはずだからです。

さて，私たちはなぜ外国語を学ぶのでしょう。必修科目だから，では身も蓋もありません。おそらくは外国人と話すため，外国人の書いたものを理解するためなのでしょう。外国人に対して自分の考えを伝えるときや外国人の考えを読み取るときに外国語が必要になるのです。

ただし，人間の考えは似たり寄ったりとはいえ，文化や風習の違いによって，その思考は少しずつ異なってきます。それは伝えたいことに対する言語での表現の差になって表れてきます。たとえば，自分の名前を名乗るとき，英語では，

　　My name is Akira Nakamori.
　　（私の名前は中森章です）

ですが，ドイツ語では，

　　Ich hei*β*e Akira Nakamori.
　　（私は中森章と称します）

となります。

本当に外国語を使いこなすためにはその言語の言い回しや決まり文句までも真似ることが必要なのです。よく，外国語を話すときは外国語で考えろといわれますが，それは思考が言語の表現に影響することが多いためなのです。

また，思考に用いる言語によって考えは制限されるという面もあります。たとえば，一般的に料理が大雑把なアメリカ人やイギリス人の使用する英語で，「まったり」とか「こってり」といった微妙な感覚の違いを表すのは不可能ですから，味覚に関する表現は，英語では「good」とか「delicious」といった非常に単純な表現に落ち着くのです。

プログラミング言語でもそれが生み出された背景によって言語の表現能力が決定づけられています。たとえば，数式を処理するために考案された言語であるFORTRANは数式を扱うための多彩な機能を持っていますが，非数値計算を行うための機能は貧弱です。コンピュータに常識を表現させるために考案された言語であるLISPはリスト処理などの非数値計算は得意ですが，数値計算のためには加減乗除程度の機能しか備わっていません。

このほかにも事務計算用にCOBOL，教育用にPASCAL，システム記述用にC，とある特定の応用分野を狙ったプログラミング言語が山のように存在しています。

しかし，ある応用分野を狙って作られているプログラミング言語がほかの応用分野で使用できないということはありません。よほど特殊な目的で書かれたプログラムでない限り，あるプログラミング言語で記述されたプログラムを別のプログラミング言語で記述し直すのはさほど難しいことではありません。

プログラミング言語とはある処理手順を記述するための手段でしかありません。行う処理と使用するプログラム言語の種類によって，記述したときの表現が簡潔になったり複雑になったりすることはありますが，ある特定の言語を使わないと実現できない処理というものは存在しないのです。もし，考えた処理手順を表現するだけの能力がプログラミング言語にないとしたら，その言語は欠陥品といえるでしょう。

データ構造とアルゴリズム

プログラミングとはコンピュータに処理させる手順を考えることです。プログラミング言語でプログラムを書くという作業はプログラミングではありません。それはコーディングと呼ばれる作業です。この2つは，ソフトウェア会社などでは，プログラ

ミングをする人はプログラマ，コーディングをする人はコーダと呼ばれて厳格に区別されています。

プログラマにとってはコーダがどのプログラミング言語を使ってコーディングしようがどうでもいいことです。実現方法はどうであれ，プログラマが想定している目的が達成できればよいのですから。

プログラミングとはデータ構造とアルゴリズムだということは昔からよくいわれます。これはまぎれもない真実で，プログラミングの本質を非常に簡潔に表している言葉です。つまり，データ構造を決めて，そのデータを処理する手順を決めればプログラミングという作業は終わりです。

データ構造とはデータの表現方法です。これは，コンピュータに処理させようとしている問題を，コンピュータは処理しやすいように記号化（あるいはモデル化）することです。

具体的には，これは受験番号に対応して成績を記録する配列を用意するとか，ある変数の値が1のときは対応するデータの値が有効であることにするという，データに意味づけを行う作業になります。

また，データ構造は，それを決定するとき，その処理方法（アルゴリズム）もある程度念頭において決定します。たとえば，配列を使用するのは添字でデータを参照することを念頭に置いているからにほかなりません。使い方（参照の仕方）を考えずにデータ構造を決定したのなら，せっかく決めたデータ構造が，本当にコンピュータで処理できるかどうかわかるはずはありません。

つまり，データ構造とアルゴリズムとは，独立した要素のように見えて，実は表裏一体をなすものなのです。極論すれば，データ構造を決めればプログラミングは終了したのも同然です。

言語とデータ構造

ところで，データ構造とアルゴリズムが表裏一体であるように，データ構造とプログラミング言語も切り離して考えることはできません。これは先ほどからのプログラミングとプログラミング言語は別物という主張と矛盾します。しかし，概念的には無関係であっても，実際の思考は具体例を頭に描きながら行いますから，プログラマの頭の中には特定のプログラミング言語が存在しているものなのです。

そして，データ構造は思い描くプログラミング言語の文法（特に演算）に強い影響を受けます。数式処理言語，リスト処理言語，事務処理言語，と，昔から数多くのプログラミング言語が考案されています。これらの言語の大きな違いは主として扱うデータ構造の違いです。

結局，プログラマが頭の中に描いているプログラミング言語によって，主となるデータ構造の形式に差異が生じてくることになります。外国語と外国の文化・風習ではありませんが，プログラミング言語が扱うデータ構造の幅を狭めてしまう可能性があります。もし，プログラマがひとつのプログラミング言語しか知らないとしたら，結構悲劇的なことでしょう。プログラマはいろいろな言語を知っておいたほうが，プログラミングでのデータ構造の選択に幅を持たせることができるようになります。

大学の情報処理系の講座では，多くの場合，PASCALとLISPが必須科目となっています。PASCALは非常に多彩なデータ構造を有しているプログラミング言語であり，アルゴリズムを記述するのに適しているといわれています。

一方のLISPはすべてのデータ型をリストだけに統一したプログラミング言語であり，リストの解釈によってそれがプログラムになったりデータになったりするという独自の特色を持っています。

データ構造の扱いに関しては両極端の2つの言語ですが，これらを学ぶことによってプログラミングの勘所が養えるようになっているのでしょう。この2つの言語を知らずして，プログラミングをするのは実はとてもおこがましいことなのかもしれません。

しかし，世の中にはPASCALやLISPを知らなくても，なに不自由なくプログラムを書いている人がいます。こういった人たちは，おそらく，他人の書いたプログラムを非常にたくさん読むことによって，本来はPASCALやLISPで学ぶべきデータ構造の知識を知らず知らず身につけているのでしょう。

プログラミングの実際

プログラミングの最大の難関は対象をモデル化し，データ構造を決定することです。プログラミングが苦手な人は，例外なく（といえるほど多くの例を知っているわけではないが），対象をモデル化するのが下手な人です。逆にいえば，対象のモデル化が上手な人がプログラミングの得意な人ということになります。

それでは，プログラミングの得意な人はどのようにして対象をモデル化し，データ構造を決定しているのでしょうか。答えは簡単です。彼らはどんなデータ構造を採用したらよいか，プログラミングを行う前から，すでに「知っている」からです。つまり，たいていの問題に対しては，過去の経験から，どう対処したらよいかがわかってしまうのです。

また，彼らが扱ったことのない問題でも，過去に解いた似た問題からの類推で解決してしまうのです。このように考えてくると，プログラミングとは場数を踏むこと，という当たり前のような，泥臭い結論に落ち着きます。

プログラミングとは，問題の解法として，記憶の中にあるいちばん近い方法を思い出して変形しているだけなのかもしれません。過去の問題をどの程度まで自分で拡大して応用することができるかは個人差があると思いますが，経験に頼る部分が大きいことは否定できません。この点，プログラミングとは人から教えてもらうものではなく，自分で体験するものだといえるでしょう。大学などで解かされる練習問題，人の書いたプログラムの改造，アルゴリズム集などの教科書の例題など，すべての体験が別の新しいプログラミングのための基盤となっているのです。

おそらく，過去の体験で捉えきれないような問題をプログラミングすることは不可能です。

*　*　*

文法などは二の次です。プログラミング言語にとらわれてはいけません。アルゴリズムさえ確定すれば，どんなプログラミング言語を使おうがたいした問題ではありません。プログラミングというものは，プログラミング言語の文法を修得するよりも，これまでどういうプログラミングを行ってきたか，あるいはどういうデータ構造をどのように使用してきたかという体験のほうが重要です。

プログラミング言語で記述された結果としてのプログラムをなぞるより，そのプログラムで行おうとしている「精神」を読み取ることが肝要です。いい換えれば，与えられた問題に対して，どのようなデータ構造を使い，どのようなアルゴリズムで解決しているのかを読み取る練習こそが必要なのです。こういった態度こそ，プログラミングの上達の最短コースではないかと思います。

フローチャートによるアイデアのまとめ

プログラムの流れをつかもう

Fuzuki Ryo 文月 涼

なんらかのプログラムを作るときに，効率よく作業するには「とにかくまとめるクセをつけておくこと」が大切です。そして，プログラムの流れをまとめるために存在するのがフローチャートなのです。

私たちをとりまく環境のなかにはいろいろなメディアがあります。ごく一般的なものとしては，新聞・ラジオ・テレビなどなど。これらのメディアを分類してみると，それぞれ自分から能動的にそのメディアに接近する度合いが違うことがわかります。

たとえば新聞はテレビなどと比較してそのメディアから情報を得るために，かなりの能動的行動を要します。その代わり，新聞を読んでいるのとテレビを見ているのとでは，得た情報の脳内での保存率が著しく異なります。つまり，自分が能動的な状態でメディアに接近すればするほど，その情報によって脳を刺激する率が高く，古いいい方をすれば脳のしわが増えるのです。

さて本題のパソコンのお話。パソコンは2種類の面を持っていて，ひとつはゲームなどのアプリケーションを使用するための面，もうひとつは自分がほしいと思うプログラムを容易に作成できるという面がそれです。一方は受動的メディア，もう一方は能動的メディアの属性です。

いまどきのパソコンはゲーム専用機と同じで，ゲームをやるだけなら電源一発なのですが，自分で作りたいものを作れるという開放されたメディアという面を持っていることは注目すべきでしょう。この点においてはパソコンは受動的メディアでなく，能動的に接してはじめてより多くの深い刺激をもたらすメディアなのです。

このような開放されたメディアを持っているのに，使わない手はありません。そして，あなたが能動的にパソコンに接すれば，パソコンとあなたの関係はより親密なものになるでしょう。

なにを作ろう

プログラムを作る動機はなんでしょう。先日某所の凄腕プログラマに聞いたことがあります。彼の答は「そこにX68000があるからだ」でした。うーん人間が練れている，と私は思ってしまいました。

まぁ，別段動機に関しては深く考える必要はないでしょう。あなたの生活に空気のようにX68000がとけ込んでいるならば，絵を描いたり，ご飯を作ったりする感覚で十分です。要は「やろう」っていう気持ちです。必要なのはただそれだけ。開放されたメディアでは，作りたいものを作るという行為が許されているからです。

プログラミングによってなにが得られるのでしょう。第一にプログラムそのものです(あっ石を投げないで)。そしてコンピュータの思考を理解するアイデア。そのほか，ハードウェアへの征服感，プログラムを発表し評価してもらう喜び，などなど。

私自身の場合，プログラミングを通じて得た感覚で，現在とっても重宝しているのは「コンピュータの思考回路を理解するアイデア」です。私は副業として自動車会社に勤務しシステムの仕様を作成していましたが，実際に作業をするプログラマの方との打ち合わせ席でも，この感覚のおかげで労力が少なくてすみました。

ま，実生活の効用はおまけみたいなもんで，ホビーとしての効用は，プログラミングを通してパソコンへの理解がより深まっていくことでしょう。自分のパソコンが自分だけのパソコンへと変わっていくのはいいものですね。では，どうすれば親密になれるのでしょう。

やってみましょう

亜希子ちゃんは13歳です。みんなには詰めて「アッコちゃん」と呼ばれています。顔立ちがチャイニーズなもんで，お団子ヘアにするとストIIの春麗にちょっと似ています。

アッコちゃんのパパは修一郎さんといいます。パパはよく家にいて絵を描いているのですが，アッコちゃんにはパパがなにをやっている人なのか難しくてわかりません。そういうわけで，友達に「パパのお仕事は？」と聞かれると，「芸術家」と答えることにしています。

アッコちゃんとパパは散歩の途中，よくママに内緒でゲームセンターに行きます。パパはとってもゲームがうまいので，つられてアッコちゃんも上手になりました。

そのアッコちゃんが最近ゲームセンターのゲームに飽きてしまったのか，「こんなゲームがあったらええのにな」といっていたのを聞いて，パパは夏休みに入るとアッコちゃんに，押入にしまっていた「パソコン」をくれました。

パソコンの名前は「AppleII」といい，パパが昔使っていたパソコンだそうです（私が初めて使ったパソコンです)。アッコちゃんはパパの書斎にある，よくしゃべる黒いパソコンを使いたかったのですが，だめ，といわれてしまいました。理由を聞くと「このAppleIIいうのんは，いまのパソコンのようにごてごてした服を着てへんし，パソコン本来の姿に近いんヤ」といいます。アッコちゃんは「けち」と思いました。

AppleIIはわりとおっきな白い箱で，箱にはじかにキーボードがついています。ほかにはボリュームみたいなノブとボタンが1個ついたパドルが計2個ついていました。

パパはパソコンと一緒にテープに入ったゲームを2本くれました。「ブロック崩し」と「なんか」です。「なんか」は英語がたくさん出てくるので，英語を習い始めたばかりのアッコちゃんには理解できないのでほおっておくことにしました。

「ブロック崩し」はお馴染みの簡単なゲームです。飲み込みの早いアッコちゃんは，「スピニングバードキーック！（バキッ)」というわけで，あっという間にブロック崩しをマスターして，次なるゲームを求めてパパの書斎に行きましたが，パパは自分が

遊ぶための麻雀ゲームのほかはなにも持っていませんでした。しかたなくアッコちゃんはパパが教えてくれた街のパソコン屋に行ってみましたが，見たところテープのゲームはおいてありませんでした。

家に帰ってパパにその話をすると「もう古いパソコンやからなぁ，ないかもしれへんなぁ」といったまま，仕事が忙しいらしく，それ以上取り合ってくれません。アッコちゃんはふてくされて自分の部屋に帰るとその日は眠ってしまいました。

翌日起きると，アッコちゃんの机の上には埃をかぶった本が数冊おいてありました。パソコンとBASICのマニュアル，ゲームの作り方の本などです。そしてメモ，「ないんやったら作ったらええんや」

プログラムを始める

アッコちゃんのお勉強が始まりました。マニュアルにはパソコンとおつきあいする作法や基本的な操作の説明，BASICの本にはパソコンを使ってプログラムを作るためのさまざまな命令の解説と実際にコマンドを使ったプログラムの例が書いてあり，アッコちゃんはそれを順番にこなしていきます。柔らか頭のアッコちゃんはそうして基礎編を数日でマスターしてしまいました。「画面に字を書いて」や「どうするか，きめて」をはじめ，さまざまな命令を覚えていくうち，アッコちゃんは気づきます。

「この命令をいろいろ組み合わしたら，いろんなことができるのかもしれへん……」

するとアッコちゃんのなかにはフツフツと野望がわいてきました。「ゲームがほしい，それもうんと派手なやつがええなぁ」

思い立ったアッコちゃんは「超ド派手シューティングゲーム」を目指してプログラミングを始めました。しかしあふれでる想像力の産物を画面で具体化しようとすると，超えなければならないいくつかのハードルに気づきます。

たとえば画面上に絵を描く方法。字の書き方は覚えましたが，画面上にブロック崩しのような絵を描くには，どうしたらいいのかわかりません。加えて，ゲームはキーの入力があるまで，勝手に敵がうろうろしていなければならないのですが，キーボードに入力を求めると，キーが押されるまでプログラムが前に進まないのです。

そこにパパが様子を見に現れます。

「もうなんか作ってるんかいなぁ。エらいなぁ」

「うん。でもようわかれへんことがあって，うまくいけへんねン」

「ふぅん。マニュアル読んだんかいな？」

「読んだョ」

「どこらへんまでや？」

「基礎ンとこ」

「で，なに作っとんのン」

「この前パパと行ったゲーセンにあるような，バリバリのシューティングゲームや」

パパはびっくりしたような顔をしたあとに，ニヤッと笑って「マニュアル最後まで読んでミ。それと，なんか作る前には，どんなことがしたいんか考えをまとめて，紙に書いたらわしンとこ持ってき」といったまま部屋を出ていってしまいました。

しかたなくアッコちゃんがマニュアルをさらに読み進めていくと，文字を表示するのが得意な画面と，絵を表示するのが得意な画面があるのがわかりました。この絵を表示するのが得意な画面をグラフィック画面といい，文字を表示するのが得意な画面をテキスト画面という名前なのも覚えました。そして人間が実際に見るディスプレイの画面にはこの複数の画面が重なって表示されているのも。

さらにわかったことはグラフィック画面を使用すると，絵の画面が字の画面の前に出てきてしまうために，字が見えなくなってしまうのと*1，もうひとつ，いちいちキーを押さなければゲームが進まないのを避けるには，キーが押されてなくても，対象とするキーが常にデータを発しているか，入力を求めた時点で入力がなければ無視して先に進んでくれるようになればよいわけで，それにはパドルに対して入力の有無を確認するればいいということでした。特にパドルのボタンを使用する方法は，キーが押されていようがなかろうが，勝手にゲームは進んでくれるという，シューティングゲームには不可欠な要素です。

さて，命令も覚え，ゲーム用のネタとなるテクニックが揃ったところで，どのようなゲームを作成しようか内容を詰めることになります。

テーマは「ゲーセンでやったゲームみたいに奥から敵が飛んでくるシューティング」です。スタートすると，画面の奥からだんだん敵が近づいてきますので，２個のパドルを使って照準を合わせ，パドルのボタンを押すと照準に向かってビームが飛びます。撃った弾が相手に当たれば敵は撃墜，当たらなければ敵は逃げていってしまいます。一定の数の敵を撃破できればおしまい。それまではずっとゲームが続きます。

アッコちゃんは決めたゲーム内容を紙にまとめると，パパのところに持っていきました(表１参照)。パパはアッコちゃんが書いたメモを見ると，フンフンと頷き，白紙を出してきてなにか描き始めました。

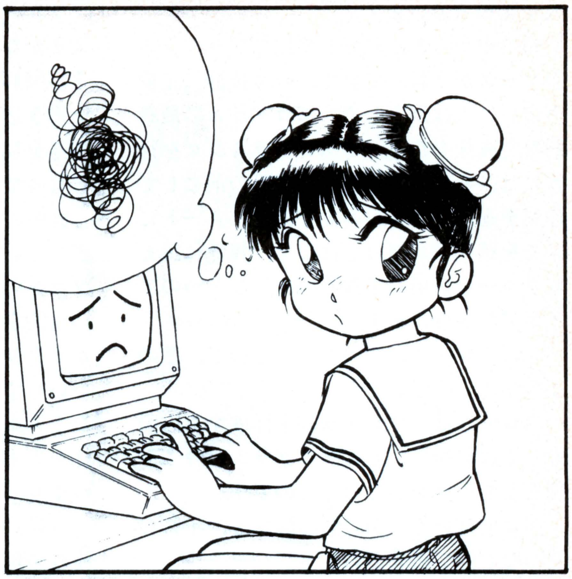


illustration:K.Tsuchida

*1　グラフィック画面がテキスト画面の前にあるかどうかは，各機種のハードウァアの構造に依存します。

フローチャートのお話

プログラミングを始めるにあたって私がおすすめするのがフローチャートです（以下フロー）。なに，そんなもの実際に描いてる奴ぁいないですって。いいんですか？プログラムを描いているうちに大きくなりすぎて，なにをやっているのかわからなくなった人や，「昨日の私は他人さ」と，自分がかつて作ったプログラムがどのような構造であったかを忘れてしまったプログラマを，私はたくさん知っています。

また，多人数で作業を分担して作成したのいいのですが，全体的構造を意識して統一していなかったために，プログラムを集合体とする段階でスリ合わせに苦労した

表１：アッコちゃんのメモ

１：ゲームをスタートする
２：パッドの数値を読み取って照準を描く
３：用意したデータに従って敵を描く
４：パッドのボタンが押されていたらビームを撃って，敵に当たっていたら撃墜
５：一定数の敵を倒すまでゲームは続く

人々も知っています。

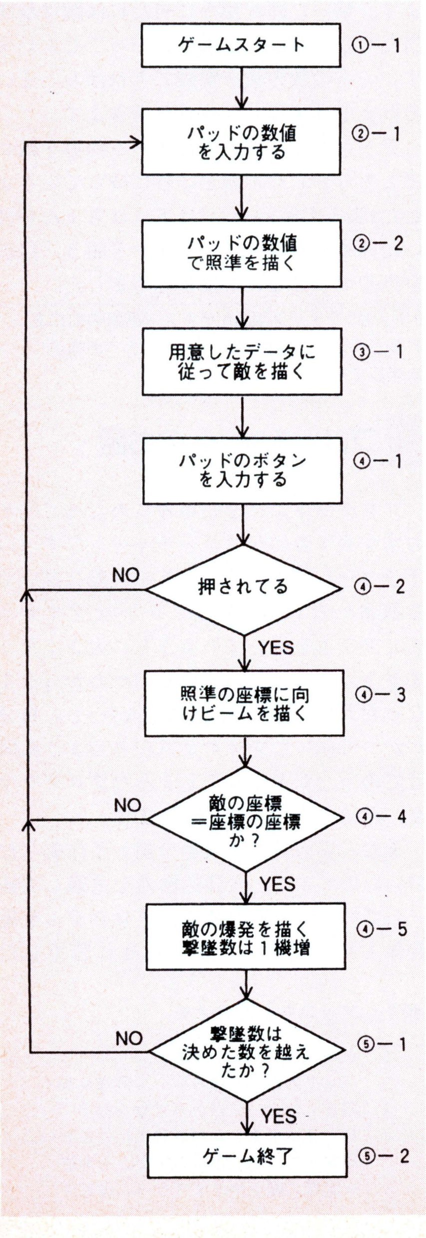
頭の中からあふれでる感性でプログラムのソースがうねっていて，かつ抜群に記憶力のいい人は別として，プログラムの最終的な完成像を意識してプログラミングを行うには，その完成像を実際に目の前にして作業を進めることが有効であり，ノートにメモを羅列するだけではなく，視覚に訴えるフローを併用してみるのは，ひとつのアイデアだと思います。

ではフローとはどのようなもので，どのように書くのでしょう。

フローとは，プログラムを作成する以前もしくは以後に，そのプログラムの骨子を，図として書き記しておくものです。

図は通常いくつかの形の図形と，それらをつなぐ線で構成されます。

図1：初期の簡単なフロー



フローの中の各記号は，コンピュータの大まかな「行動」の性格を，記号の中にはその図形の段階で行われている処理の概略を書き込み，線は処理の流れを表し，その構造を視覚的にとらえることができるようになっています。

図中の記号はコンピュータの「行動」を表すといいましたが，「行動」はここではパソコンの処理の単位を意味し，必ずしもその量的定義があるわけではありません。細かく書きたい方は1命令単位で作成してもいいですし，多くのサブルーチンの集合となるメインのプログラムであれば，その仕様を大まかにまとめたものでいいのです。つまり「行動＝1命令」とは限りません。

「行動」の単位を決定すると，次の作業は流れの規定となります。この「行動」をさせたあとに，あの「行動」をさせるというようにです。並び方を決定し，流れを表す線で記号を結び，フローは完成します。

要は，作成しようとするプログラムの全体像をつかむための手法なのです。

ではその「行動」の単位はどのように決定するのでしょう。パパの話の続きを聞いてみましょう。

「アッコはパソコンに命令する仕方をようけ覚えたんやけども，いきなりゲームを打ち込み始めてもダメなんや。最初に完成するプログラムをイメージしておかないと，だんだん収拾がつかんなってあとで痛い目を見るんや。たとえば夏休みの宿題みたいにナァ」アッコちょんはパパにそういわれて，ちょっと舌を出しておどけました。パパは笑いながらアッコちょんのメモを書き直して番号をふり，机の上にあったいろんな穴のあいた定規*2を使って，四角やひし形を並べ，さっさかと図を書きました（図1参照)。アッコちょんは珍しそうにその手元をのぞき込んでいます。パパは書き終わると，紙をアッコちょんのほうに向けてくれました。

「パソコンの行動っていうんは，だいたい4つぐらいに分けることができるんや。ひとつ目が，パソコンが自分の外からデータを持ってくる『入力』。2つ目がパソコンがテレビなんかに絵や字を描く『出力』。3つ目はパソコンが自分の中で足し算をしたり掛算をしたりする『処理』。そして，パソコンがいまあるデータに従って，どういう行動をしようか決める『判断』や。この，4つを覚えたら，図を見てみ。図の中では2つの図形しか使ってへんけど，四角形で中に『入力』と書いてあるんが『入力』の図形や。『出力』は四角形で中に『描く』と書いてあるのんや。『入力』も『出力』もない四角形が『処理』。ひし形が『判断』や。メモの横にふってある丸つき数字と，図形の横にふってある丸つき数字がそれぞれ対応していて，アッコの作ったゲームの構想をパソコンにわかりやすいようにすると，だいたいこれぐらいに分けられるんや」

アッコちょんはふうんといいながら，メモと図を見比べています。

「プログラムを作るときは，こうやって自分の考えをパソコンにわかりやすいように書いて紙にまとめておくとエエんや。こうするとプログラム全体の見通しもいいし，行動の順番も一目瞭然。あとになってプログラムの構造を思い出すのも簡単やろ。もっともっとプログラムを作るんがうまなったら，大まかに書いても大丈夫やろうけど，最初のうちはいいつけを守って細かく書いとき。わかったか」

アッコちょんはわかる範囲で理解すると大きく頭を縦にふりました。

「ようし，ほしたら作ってミ。困ってもやたらとワシを呼んだりせんと，マニュアルを読み返すんや。本は何べんも読まんと頭

アッコちょんのパソコン

文中でアッコちょんが使用していたパソコンのモデルは，米アップル社の「AppleII J-plus」です。いま考えるととても先進的な思想で作られたマシンで，本体の上部のカバーをはずすと，その中には底に張りついた基板が1枚だけあって，ほかにはなにもなかったのです。ではそのほかのスペースはといいますと，確か7〜8枚ぐらいのスロットが用意されて，そのスロットをメモリ増設とか，フロッピーディスク基板とか，デジタイザタブレット（のようなもの？）をつけるために使用するのでした。また，標準でついてきたパッドは，なんとアナログだったのです。

しかも文中で出てきた『定義形状』に相当する機能があり，プログラムの中で，一筆描きで定義した形を，拡大縮小回転させて画面に表示することができたのです。これだけの機種が約12年ぐらい前に存在していたとは驚くべきことで，しかもソフトウェアとして，フライトシミュレータ（ポリゴン）も存在していたことを考えると，AppleIIが先進的だったのか，いまのマシンが進歩していないのだか錯覚してしまいます。

標準でAppleIIについてきたもうひとつのゲームである「なにか」は「レモネード」といって，毎日レモネードを作って売るゲームなのですが，前日に天気予報を見て仕込みをし，翌日の天気によって売れる数が決まるという，スロットマシンにも似た感覚に，わくわくしたものです。

に入れへんし，自分ひとりで一生懸命考えるっていうことを覚えてみ」
「うん」
「それと，命令の使い方なぁ。マニュアルに書いてある使い方だけではなくて，別の使い方を考えてみ。コロンブスの卵や」
「……？」
「そのうちわかるって」と，まぁこんなふうにパパはアッコちょんに説明したのですが，ちょっと補足しておきましょう。

４つの行動の分類に関しては，異論のある方もいらっしゃるでしょうが，ここではひとつのアイデアと考えてください。また，実際のフローチャートは図のように２種類の記号だけではなく，多くの記号で作成されますが，ここでは簡略化するために２個のみを使用しました。

さて，図中のフローの各記号はコンピューターの「行動」の単位を示していますが，実際のプログラミングの際には，各種の言語ごとに用意されている命令群を用いて「行動」を構成します。

この「行動」をパズルとたとえると，各言語ごとに用意された命令群はそのパズルを完成させるためのコマと考えられます。ただ本当のパズルと異なる点はパズルのコマに限りがないということです。ひとつの「行動」を作るのにいくつコマを使ってもよいのです。よりよい順番でコマを並べればコマの数は少なくてすみますし，使うコマが少ないと，プログラムを作る人間にとっても，それを実行するコンピュータにとってもラクチンになるのです。

アッコちょんのプログラミングが始まりました。最初の頃は例の「ピー」と「プー」ばっかりだったのですが，マニュアルと首っ引きで悪戦苦闘し，バグ取りをして，なんとかそれらしきものができました。

ひと通り作り終わってプログラムを走らせるてみると，続々と問題点も出てきます。まず，画面上に描いた図形のすべてがそのまま残ってしまうというもの。照準を動かしていくと，その形で画面を塗り塗りしてしまうのです。敵もさるもの，一緒になって塗り塗りしています。これは，一度照準を描いたあとに，もう一度背景の色で照準を塗ることで解決しました。

次は，敵が早く動きすぎて照準で追いきれないという現象です。これは１回の処理ループごとに敵を動かすのではなく，何回かおきに敵を動かすことにしました。

ある程度の完成を見たところで，アッコちょんはパパを呼んできてゲームを見せてみました。パパはアッコちょんの説明を聞き終わると，ニヤっと笑い，ゲームをプレイし，次にゲームのリストを見始めます。アッコちょんはそのあいだ，画面を滑るパパの指と顔をかわるがわる見ています。
「ひと通りできたみたいやな。……ん，なんか聞きたそうやん」
「あんね，ほかのゲームもこないな風にしてできたん？」
「せや」
「ふぅん」
「ゲームだけやないで。たとえば自動販売機や，ママがつことるファジィの洗濯機やって，結局はアッコがやったんと同じように，誰かが作ったプログラムで動いとるんや。そのプログラムを細かく見てみると，結局は１つひとつの命令の組み合わせなんや」
「へぇ」アッコちょんは感心したようすで，部屋の中にあるいろんな機械をながめています。パパはまたニヤッっと笑って「もっといろんなこと知りたいか」といいました。アッコちょんは大きく頷きます。

＊2　フローチャート用の定規は，たいていのコンピュータ屋に行けばおいているでしょう。フローチャートの定規といってわからない場合は，「いろんな形の穴があいている定規」といえばわかるんじゃないでしょうか。定規には，穴の横にそれぞれの図形の名称が書いてあります。参考にしてより綺麗なフローチャートを書いてみてくださいね。

手を加えてみる（最適化へ）

さて，一応プログラムは完成したようですが，まだまだ終わりません。
「アッコはゲーム作ってみて，ある程度パソコンとお話しできるようになったよな」
「うん」
「これでおしまいにするか」
「ううん」
「ほしたら，次はゲームをバリバリにせな」
「うん！」

パパはBASICのマニュアルの応用編のページを開きました。「いまのアッコのプログラムでは，敵はいくつかのパターンを持っとって，それを順繰りになぞってるだけや。これではおもんない。もうひとつは，いまは同じ形の敵がパターンどおりに動き回っているだけや。これもおもんない。せやから，まずこの２つに絞ってバリバリにしよな」
「うん」
「まず，順繰りに出てくる敵やけど，パターンを用意するのはかまへんねん。せやけどそのパターンが順繰りに出てくると，順番を覚えられたらすぐにおもんなくなるで。ここは『乱数』を使うんや」
「乱数ぅ？」

アッコちょんは首をかしげ頭の中を探してみました。
「マニュアル読んだやろ」
「うん。せやけど，どんな使い方するんかわからへんかってん」
「コロンブスの卵や。敵が出てくるところで乱数を発生させて，用意してるパターンのどれかを選ぶんや」
「うん」
「動きを用意するんはしかたないけど，出てくる順番は誰にもわかれへんなるやん」
「うん」
「もうひとつは敵の形や」

そういってパパはマニュアルのあるページを指しました。そこには『定義形状』の説明が書いてあります。
「今の敵は，アッコが決めた形で上下左右を動くだけやけど，たとえばゲーセンのゲームやったら，敵の飛行機がこっちに向かって飛んでくるとき，だんだん大きくなるし左右に回転するやろ」
「うん。せやけどそれって，むっちゃムズいんちゃうのん？」
「これやこれ」

パパが指した「定義形状」のページには，あらかじめ「形状」をプログラムの頭で定義しておいて，敵を画面に描くときには線

トップダウンとボトムアップ

プログラムを作る際の手法に２つあります。

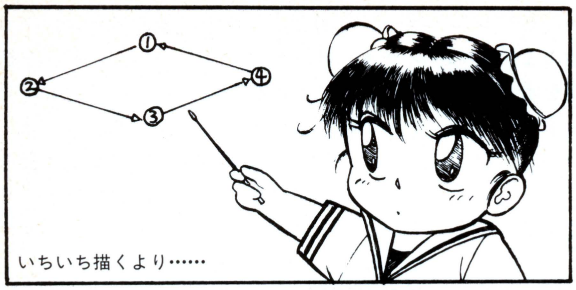
まず仕様を決め，おおまかなフローチャートを作ります。次にフローチャートを描き，そしてそのなかのひとつの処理にまた，……とプログラムの１命令になるまで繰り返すのがトップダウン式のアプローチです。

逆に，いくつかの簡単なサブルーチンを作り，そのサブルーチンを使ってもっと大きなサブルーチンを作り……というのがボトムアップ式のアプローチです。

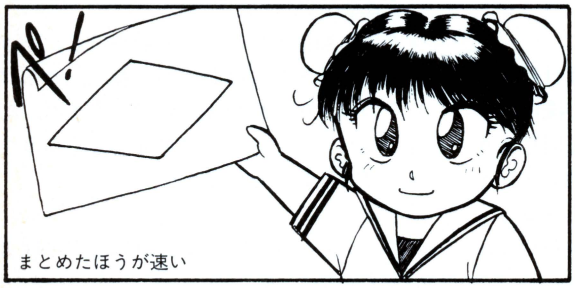
トップダウンでは途中でミスがなければ素晴らしく効率的に作業が進みます。ただし全体が完成するまでプログラムは動きません。

ボトムアップでは，各部のつじつまがあっていればきわめて広い応用のできる部品群ができあがります。ただし，サブルーチンを汎用にしなければなりませんし，そのようなプログラムは多少無駄を含んだコードになりがちです。

実際には両者を組み合わせて作業する人が多いようです。仕様を決めて，部品から仕上げる……詳細は105ページからの横内君の記事を参照してください。



いちいち描くより……


まとめたほうが速い

を描くのではなく「『形状』を描け」と命令し，パソコンは決められた「形状」を使って画面にさまざまな形を描く，と書いてありました。
「ほしたら，これ使うたら，敵を描くとき『どの点とどの点とどの点を結んで』っていわんと，『飛行機の形を描いて』っていうだけでええのん？」
パパは頷いてこう答えました。「それだけやあれへんで。こいつは賢いから『飛行機の大きさが，元の何倍，何度回転させてから描いて』っていうのもできる」
「照準も？」
「そう。ようわかったナ。それに，たとえば字の形を登録しておいたら，アッコがいってた『グラフィック画面に字を描く』のもできんで」
「ほんまぁ」
パパは感心しているアッコちょんを見てちょっとうれしそうです。
「命令の使い方にもいくつか方法があって，プログラムの行数を減らして速くする方法もあれば，行数は変われへんでも派手な命令を使ってゲームを面白うする方法もある。形状の場合は両方を満たしとるな。ゲームのスピードは速なるし，ゲームもバリバリになるし，プログラムの行数も減るし，いいことずくめや」
「うん」
「うれしヘルシーやろ」
アッコちょんは頭を抱えてしまいました。
アッコちょんはプログラムのバージョンアップを始めました。基本的にはフローに書いたプログラムの骨子は変更しないで，いろいろ手を加えていくのです。
大きく変わった点は，乱数と定義形状の利用です。まず，乱数は敵の出現パターンの選択のときと，出てくる敵の形状の選択のとき，そしてパターン上を動くときにも，乱数でちょっとずつ上下左右にずらすようにしました。
定義形状は，照準，敵の形，撃墜数の表示で使用しました。またアッコちょんが自分のアイデアでやった変更は，音の出し方を覚えたので，敵を撃ったときや撃墜したときに，ピッピと音を鳴らすようにしたのと，ゲームスタートのときにプレイヤーに目標撃墜数と難易度を入力してもらい，それに従ってゲームを進行させるようにしたことです。
さらにこのあと細かいバグ取りを行い，アッコちょんのゲームは完成を見ました。
アッコちょんはその後，遊びにくる友達にさかんに自分が作ったゲームをすすめて，意見を聞いては，手を加えているのです。

プログラムってな〜に

「プログラムってな〜んや」
「プログラムってな，パソコンを使っていろんなもんを作ることや。パソコンって自分が作りたいナァと思ったものを，作ることができんネン」
「アッコにとってのプログラムちゅうんはそうやね。ムズかった？」
「最初はパソコンがなに考えてるんかわかれへんで，ごっつムズかったけど，ゲームを作ってるうちに，パソコンがどんな考え方をしてるんかわかってきたワ」
「パソコンの考え方は，結局はいくつかの基本的な命令の組み合わせなんや。それをわかってると，ほかのコンピュータをいじるときもそんなに困れへんのや」
「ふぅん」
アッコちょんはまた大きな目を開いて頷いてみせました。
「今回，アッコはパソコンの使い方を勉強したし，少しパソコンとのおしゃべりの仕方も覚えたなぁ」
「うん」
「ほしたら，今度書斎に置いたあるパソコンを使わしたるワ」
「あの，黒いのん，触ってもええん？」
アッコちょんは，にこにこしながらうれしそうに手をたたくのでした。

まとめてみましょ

さて，アッコちょんはこうしてパソコンとのおつきあいを始めたのですが，彼女のように，柔らか頭でパソコンとの会話ができるようになるのは，ちょっとした才能というより，まだ頭の固まっていない若い人の特権でしょう。
では才能のない人にはプログラムを作成できないのでしょうか。そうでもありません。なぜなら，プログラムを作成すること自体はそれほど難しいことではないからです。プログラミングというと，しりごみをする人の話をよく聞きますが，それは決して難しいものではありません。あえて難関というものが存在するとするならば，それは自分の思考回路と異なった考え方を持つものに対する気後れでしょう。これは別段非難されることはなく，生物全般の自然な性質でしょう。
最初にも書きましたとおり，パソコンは現代的な受動的風習，すなわち受け手として流れてくるもののみを居ながらにして受け取ることの多い環境のなか，数少ない双方向の交流が開放されたメディアといえるでしょう。せっかくパソコンを持っていながらこれを使わない手はありません。
プログラミングを通じて得られるのは，もっとパソコンに近づいた自分ですし，またそれを発表することによって，人と意見を交わし，さらにパソコンに対する理解を深める行動は，ひいては自分自身にまた違った一面から光を当てることになるのです。ね，悪い気分はしないでしょう。
この夏休み，プログラミングを通してパソコンとより親密になるのを課題のひとつとしてみてはいかがでしょう。アッコちょんも陰ながら応援しています。

文月とR間G郎氏の怪しい関係

今回のイラストを担当していただいたR間G郎こと土田氏は，知る人ぞ知る，ゲームデザイナー兼CGアーチスト兼おたプーの人です。彼の身近な作品を挙げますと，ゲームでは「STAR WARS」，CGでは「芸術祭のオープニング」です。炎のゲーマーでもあるらしく，編集部にスト2'の基板があると聞いたとたん，中指を立てて殴りこむつもりになったようです。氏は私が連載を持つなら「イラストを描いてやる」といっていたので，もし彼のイラストをコンスタントに見たい方は，編集部まで圧力をかけてください。イラストだけっていうのは，なしよ。

X-BASICで書くとこんな感じ?

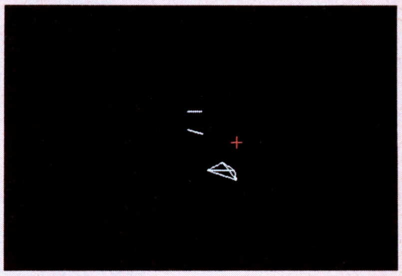
文月氏のサンプルプログラムというのは，氏が昔使っていたApple IIを対象にしている。サンプル自体も文月氏が昔作ったゲームをモデルにしているらしいが，すでにApple IIなど見たこともないという人も多いはずである。無論，プログラムを制作するにあたっての基本的な部分は機種を問わないものなのだが，これではあんまりなので，アッコちょんの作成したメモを基に X68000でプログラムを作ってみた。

X68000にはパドルがないので入力はマウスに変更している。Apple IIの標準機能（といってもソフトウェアでの）だったグラフィックの拡大・回転などはX68000ではサポートされていないので，ここではMAGIC.FNCを用いた。MAGIC.FNCは1991年7月号で発表されたX-BASIC用の外部関数である。まことに申しわけないが，MAGIC.FNCをお持ちの方でないと，このプログラムは利用できないので注意してほしい。

さて，フローチャートを見るとメインループからの出口が複数あって私の好みじゃないので多少アレンジした。130〜300までがメインプログラム。定義された関数はほとんどが手続きとして使われている。ゲームならこんなに細分化しないほうがいいのだが，いずれはコンパイラにかけるだろうこと，そして本当に賢いコンパイラならどう書いても同じコードを出すはずだという信念に基づいてこんなことをしている。

せっかくMAGICを使っているんだから，パターンの1軸回転だけではもったいないので疑似3Dではなく3D処理に変更されている。

当たり判定は「ビームと敵がすれちがったときのX，Y相対座標の和」という，ややいい加減な方法で行った。視点固定のままビームを3次元に放射しているので命中率は低い。アッパースイングでは大リーグボール3号に当たりにくいのと同じ理屈だ。まだゲーム性は考えられてないので面白くはない。保証する。(S.N.)



リスト

```
  10 /*      亜希子ちゃんのプログラムをX68000上で
  20 /*      再現してみたものSION風味
  30 /*                             (要magic.fnc)
  40 /*                            designed  by  R.F.
  50 /*                            programed by  S.N.
  60 int gekitui,border=10,teki,tama,shot,bullet=-9999
  70 int kuru,dx,dy,x_,y_,true=-1,nf
  80 int i,j,k,l,x=256,y=256,z,cx,cy,cz,vx,vy
  90 int a(99),b(99)                  :/* 形状用  */
 100 char pat(255)
 110 /*
 120 /* main program
 130 initialize()                     :/* 初期化 */
 140 /*input"難易度の設定";border
 150 repeat
 160   mspos(x,y)                     :/* マウス座標を読む
 170   scope(x,y)                     :/* 照準を描く
 180   teki_move()                    :/* 敵を動かす
 190   msstat(1,1,shot,1)             :/* 左ボタンを読む
 200   if shot=true and bullet<-100 then shoot()
 210   beam()
 220   if strike()=true then {
 230        gekitui=gekitui+1
 240        bakuhatu()
 250        z=2200
 260   }
 270   disp()
 280 until gekitui > border
 290 gameover()
 300 end
 310 /*
 320 /* subroutine
 330 /*
 340 func initialize()
 350   screen 1,3,1,1
 360   msarea(255,255,256,256)    :/* マウスを真ん中へ
 370   mouse(4):mouse(2)
 380   msarea(10,10,495,495)      :/* きいてないみたい?
 390   magic_flush()
 400   magic_init()
 410   magic_screen(1)
 420   magic_color(254)
 430   console 0,31,0
 440   /* 形状定義 */
 450   a={4,
 460       0,-20,0,
 470       0,10,-80,
 480       34,10,16,
 490     -34,10,16,
 500       6,
 510       0,1,
 520       0,2,
 530       0,3,
 540       1,2,
 550       2,3,
 560       3,1
 570       }
 580   b={4,
 590     -19,0,0,
 600     -19,0,80,
 610      19,0,0,
 620      19,0,80,
 630      2,
 640      0,1,
 650      2,3
 660     }
 670   pat={0,0,0,0,0,0,0,5,0,0,0,0,0,0,0,0
 680       , 0,0,0,0,0,0,0,5,0,0,0,0,0,0,0,0
 690       , 0,0,0,0,0,0,0,5,0,0,0,0,0,0,0,0
 700       , 0,0,0,0,0,0,0,5,0,0,0,0,0,0,0,0
 710       , 0,0,0,0,0,0,0,5,0,0,0,0,0,0,0,0
 720       , 0,0,0,0,0,0,0,5,0,0,0,0,0,0,0,0
 730       , 0,0,0,0,0,0,0,5,0,0,0,0,0,0,0,0
 740       , 0,0,5,5,5,5,5,5,5,5,5,5,5,0,0,0
 750       , 0,0,0,0,0,0,0,5,0,0,0,0,0,0,0,0
 760       , 0,0,0,0,0,0,0,5,0,0,0,0,0,0,0,0
 770       , 0,0,0,0,0,0,0,5,0,0,0,0,0,0,0,0
 780       , 0,0,0,0,0,0,0,5,0,0,0,0,0,0,0,0
 790       , 0,0,0,0,0,0,0,5,0,0,0,0,0,0,0,0
 800       , 0,0,0,0,0,0,0,5,0,0,0,0,0,0,0,0
 810       , 0,0,0,0,0,0,0,5,0,0,0,0,0,0,0,0
 820       , 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 830       }
 840   sp_init()
 850   sp_def(0,pat)
 860   sp_disp(1)
 870   teki=magic_putbuf(1,a)
 880   tama=magic_putbuf(1,b)
 890 endfunc
 900 func scope(x,y)
 910   sp_move(0,x,y,0)
 920 endfunc
 930 func disp()
 940   locate 60,0:print gekitui
 950   magic_disp()
 960 endfunc
 970 func teki_move()
 980 z=z-2100
 990   z=((z- 4) mod 2000)+2100
1000   vx=vx+1
1010   vy=vy+1
1020   cx=cx+sin(vx/30#)*7
1030   cy=cy+cos(vy/25#)*11
1040   magic_seek(1,teki,0)
1050   magic_data(1)
1060   magic_para(0,cx)
1070   magic_para(1,cy)
1080   magic_para(2,z)
1090   magic_para(8,0)
1100   magic_pers()
1110 endfunc
1120 func beam()
1130   bullet=bullet+40
1140   kuru=kuru+16
1150   if bullet<1600 then {
1160   x_=x_+dx/8#
1170   y_=y_+dy/8#
1180   magic_seek(1,tama,0)
1190   magic_data(1)
1200   magic_para(0,x_)
1210   magic_para(1,y_)
1220   magic_para(2,bullet)
1230   magic_para(8,kuru)
1240   magic_pers()
1250   } else bullet=-9999 :kuru=0 :shot=0
1260 endfunc
1270 func shoot()
1280   bullet=-100
1290   dx=x-256
1300   dy=y-256
1310   x_=0:y_=0
1320   nf=true
1330 endfunc
1340 func strike()
1350   int r=0
1360   if nf=true then {
1370      if bullet>z then {    /*      ↓当たり判定(大甘)
1380         if abs(x-x_)+abs(y-y_) < 250 then {
1390            r=true          /*      でもムズい?
1400            nf=0            /*
1410         }
1420      }
1430   }
1440   return(r)
1450 endfunc
1460 func bakuhatu()
1470   beep
1480   /* 爆発パターンは都合により省略します
1490 endfunc
1500 func gameover()
1510  locate 30,20:print"おめでとう"
1520 endfunc
```

BASICで作るDōGAフレームファイル

正しい花瓶の落とし方

Shibata Atsushi 柴田 淳

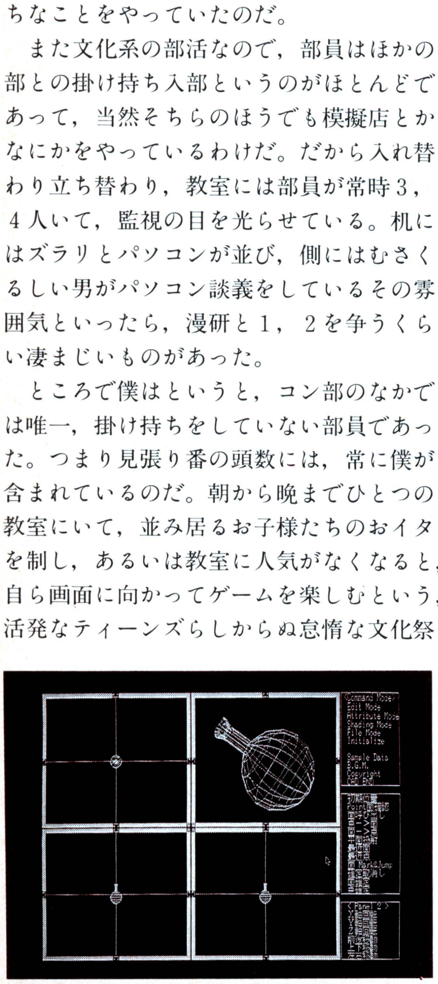
プログラミングの実践編です。人はどのようなときになにを思ってプログラムを作るのか？ どのような過程でアルゴリズムを開発するのか？ をドキュメンタリーで追ってみましょう。

高校生の頃，コン部という怪しい部活をやっていた。ただし，コン部といったって，夏になると海に出かけて海産物をどっさり採ってくるといった類のアウトドア志向の部活動ではない。コンピュータ部のことを，僕たちは略してこう呼んでいたのだ。

文化祭になると，学校の１教室を占拠して布教活動にいそしんだ。要するに，部で所有するパソコンと，リッチな奴は自分持ちの機械をきちんと持ち込んで，それぞれにゲームなどを走らせておくというありがちなことをやっていたのだ。

また文化系の部活なので，部員はほかの部との掛け持ち入部というのがほとんどであって，当然そちらのほうでも模擬店とかなにかをやっているわけだ。だから入れ替わり立ち替わり，教室には部員が常時３，４人いて，監視の目を光らせている。机にはズラリとパソコンが並び，側にはむさくるしい男がパソコン談義をしているその雰囲気といったら，漫研と１，２を争うくらい凄まじいものがあった。

ところで僕はというと，コン部のなかでは唯一，掛け持ちをしていない部員であった。つまり見張り番の頭数には，常に僕が含まれているのだ。朝から晩までひとつの教室にいて，並み居るお子様たちのおイタを制し，あるいは教室に人気がなくなると，自ら画面に向かってゲームを楽しむという，活発なティーンズらしからぬ怠惰な文化祭を僕は送っていたのだ。



これが花瓶だ

僕が高一の頃だから，いまからだいたい……いや，歳がバレるのでやめとこう。ええと，当時のゲームというのは，いまほどの派手さはなかったけれど，それでもなかにはけっこう楽しめるものがあって，そのとき展示してあったもののなかでいちばん秀逸だったのが，VIC-1001のパックマンだった。ROMカートリッジのゲームで，完璧に本物そっくりなのだ。部員のひとりが例の永パを使ってとんでもないスコアを出して，冷やかしにきた子供たちのヒーローになっていた。

ところでこのVICというのはあのコモドールのマシンなのだ。パックマンのほかにも，アーケードからの移植とか，オリジナルのとか，たくさんのゲームカートリッジが用意されていて，僕なんかこの頃からアメリカって凄いなあと思っていたのだが，よく考えるとパックマンは日本製のゲームなのだ。うーん，本当に凄いのはどちらなのだろう？

まあ，この話は本題とは関係ないので，このくらいにしておこう。ここで問題にするのは，VICの隣でデモっていた，PC-8001のクイックスなのだ。このゲームも，当時としてはなかなかよくできているほうで，当然版権などは取っていなかったのだけど，移植度は90パーセントといったところだった。しかしドットが粗いとか遅いとか，そんなことを挙げつらって90パーセントといっているのではない。このゲームの場合，その点ではむしろ頑張っているほうであった。問題は，もっと根本的なところにあって，だからこそ決定的なのだ。

クイックスというのはご存じのとおり，スティックで線を引いて，フィールドを囲み自分の陣地にするという，いってみればサイバー陣取りゲームのようなものである。囲んで陣地にした部分は，それとわかるように塗り潰される。

ところでここで，ひとつ問題が生じる。たとえばフィールドを上下に等分する場合を考えてみようか。この場合プレーヤーの陣地となるのは，上だろうか下だろうか？ この問題を解決するために，スパークといわれるジャマーが用意されている。つまり，スパークがいないほうが，プレーヤーの陣地となるのだ。ただしプレーヤーの区切る線は，この例のように単純な形ばかりとは限らない。曲がったり入り組んだりしたときにも，スパークのいないほうをきちんと塗り分けるには，ちょっとした苦労がともなう。

ところが残念なことに，このクイックスはその苦労を放棄してしまったのだ。どういうことかというと，とりあえず両方塗ってみて，塗り終わるまでにスパークに触れなければそこを陣地とするという，考えうるもっとも単純な方法で，この問題を解決してしまったのだ。だからゲームをやっていると，半分くらいの確率で陣地の反対側を塗り始め，スパークに触れるとオズオズと消し直す，といった図が展開されるのである。

で，文化際に来たガキどもが，そのクイックスをやっていて，反対側を塗り始めたのを見ると目を輝かせてこういうのだ。

「あっ，コンピュータのくせに間違ってやんの！」

これを聞いたとき，その場に居合わせた部員一同が感じたのは，悔しさ以外のなにものでもなかった。僕たちは文化際が終わると口々に，俺たちが大人になったらこんな世の中を変えてやろうと，肩を叩き合い涙にむせんだものだった。

ここで話をDōGAに移そう。いや，コンピュータグラフィック一般といってもいい。状況は変わってない。そうじゃないか？

自動車が，こう，右にカーブしながら眼前を通りすぎるとする。フレームエディタでそのような動きを描く。できあがった画

像を見てひと言。
「ドリフトしてる」
苦労して，人力でフレームファイルを修正しても，
「4WSだね」
といわれるのが精々だ。またモーションエディタで歩く人間を作っても足が地面にめり込んでしまったり，例を挙げればきりがない。
いったい，このままでいいのか？ いや，この状況は変えなければならない。だって僕は，あの日お星さまに約束したのだ。

なにが問題か考える

そもそも，なんにでもフレームエディタを使うことに問題がある。単純な中割り作業だけではこと足りない動作には，FFEを使うべきではないのだ。さっきの車の例を見ても，フレームエディタがもっとも単純な物理法則をさえ勘定に入れていない，というところに問題の根源があるようである。
もっというと，FFEだけで作ったCGAでは，因果関係が逆転しているのである。フレームファイルで扱う物体の座標とか回転角というのは，記述の方法にすぎないのだ。つまり初めにモノありきなわけで，そのモノのありさまを画面上で再現するために，記述が存在する。ところがフレームエディタを使うと，この記述の方法に物体の状態が振り回されてしまって，その結果不自然な動きが生成されるのだ。因果関係が逆転とは，そういう意味だ。
FFEの悪口ととられると嫌なのでいっておくが，これはこれで，使えるソフトだと思う。実際テレビなどのCGのほとんどは，FFEと根本思想を同じくするエディタで作られているはずだし，なにより，物体の支配される物理法則を加味して画像を生成する，なんていう重たい仕事は，CG研究の最先端でやられていることなのだ。
ではFFEを使わずに，フレームファイルをどうやって作成するのか。人力でだって？ 馬鹿いっちゃいけない。もしあなたにそんなことができるんなら，科学者とか数学者とか，もっと人類に役立つような職を目指すべきである。やっぱりここは，あなたも持っている高速計算機と高級言語を使わない手はない。
今回やることをもっと具体的にいおう。題名にもあるとおり，DōGAのフレームファイルをBASICを使って生成しようと思う。とはいっても，上から紙がヒラヒラ落ちてくるとか，地下鉄から吹き上げる風でグラマーな女性のスカートが舞い上がるとか，そんな複雑なことをやるわけではない。ただ，球体と花瓶を上から落として，ああ跳ねただの，おお転がっただのいってみるだけだ。
しかし，ただの落下する球でも，実際見てみるとけっこう新鮮な印象を受けるはずだ。繰り返しになるけど，世の中のCGというのは，ほとんどが物体の受ける物理作用を無視して作られているので，見る側は現実とそっくりな，安心して見られる映像に飢えているのである。まあ，そのことを実感してもらうには，プログラムを打ち込んでもらうしかないのだけれども。
また，あらかじめ書いておくけど，今回はDōGAのシステムについて，たとえばCADの操作方法とか，フレームファイルの書き方などといった細かいことにはほとんど触れないつもりでいる。その点でもし疑問がわけば，DōGAのマニュアルを参照していただきたい。

まずは球から探りを入れる

リスト全体で270行ほどもあるので，まずは落下する球体をシミュレートする部分だけを打ち込んでもいいかもしれない。
ball( )という関数がその部分で，ちなみに1600～2620行までは，打ち込まなくても動作に支障はない。
リストを打ち込んだら，走らせる前にCADなどを使って作っておいてほしいものが3つある。
まず，半径50の球体の形状ファイル。TAMEN.Xを使ってもよい。これが球の本体となる。“ball.SUF”というファイルネームで登録すること。
次に，これも半径50の，XY平面上にある円盤。要するに，CADの4つの画面の左上を見て，円に見えればいい。これは球の影になるので“shadow.SUF”という名前で登録。
最後は，球が跳ね返るための床。いちばん上の面が，XY平面上にあれば大きさはどうでも構わない（小さすぎるのはマズい）。ファイルネームはもちろん“floor.SUF”。なお，これらのファイルネームはフレームファイル上に物体名として書き込まれるので，変えたり間違ったりすると，フレームファイルを使って画面に書き出しをするときにエラーになる。
あと，各物体のアトリビュートについてだけど，球体と床については適当にやってもらって一向に差し障りないが，影だけは，真っ黒なアトリビュートを設定していただきたい。まあ，赤とか緑の影が好きというのなら，無理にとはいわないけれど。
準備がすべて終わったら，さっそくプログラムを走らせてみよう。いちばん初めのメニューで落下する球体を選ぶと，最初に必要な変数をいくつか聞いてくる。適当な値を入力すると，フレームファイルを書き出し始める。終わるとタイムチャートを作って，すべての作業はおしまい。DōGAのシステムを呼び出して，まずは手始めにWIREVIEWでも使って動作を確認するのがい

CADによる美しい球体の作り方

DōGAのCADは，結構便利だ。かなりよくできている。僕はDōGAを手に入れて，まずこいつでしこたま遊ばせてもらった。別になにか作りたい作品があるのでなくても，ただ遊びでオブジェクトを作って，レンダリングで描いてみるだけでも，この上なく楽しいのだ。
そんな風に遊んでいるなかで，僕が身につけたテクニックがいくつかある。そのなかのひとつをここで紹介しよう。それは綺麗な球体の作り方だ。
CADで球を作るにはどうすればいいだろうか？ 少し考えてみてほしい。
このCADには回転体を作る機能があるが，まず思いつくのがこれを使った方法であろう。しかし回転体を作る前には，元になる図形を切り出さなければならない。つまり球体を作るときは，完全な半円を切り出さなければならないわけで，これは少々厄介だ。
ここで発想の転換が必要である。この半円を切り出すために，ダミーの多角形を作っておく。正多角形作成を指定したら，カーソルを初期位置に戻し点を確定する。次にCADの画面左下の，XZ面に目を移し，作りたい球の半径分の高さまでカーソルを持っていき，また点確定。あとは同じ面上の好きな点でもうひとつ点を確定し，面確定をクリックすると目的の正多角形ができあがる。左下の画面で円に見えるはずである。
さて，せっかく作った図形だけど，こいつは即座に面削除で消してもらいたい。消したらカーソルを初期位置に戻して，XZ面で，カーソルのZ軸を作りたい球の半径まで上げる。そして「点→面」をクリックすると，なにもない空間に，先ほど消した面が点線で浮かび上がるはずである。
つまり，このCADは一度削除した面の座標をきちんと取っておいてくれる。これを利用して先ほど消した円柱をなぞれば，きれいな半円が書けるというわけだ。具体的には，中心軸をZ軸に取り，あとはさっきの要領で消した面を呼び出し，点確定したら「面→点」で隣の点を呼び出し，というのを繰り返して半円を描き，面を確定すれば，綺麗な球体のできあがりだ。
この，一度作った面を消してというテクニックは意外に応用範囲が広く，また最近点呼び出しなどと組み合わせれば，CADの操作効率は格段に上がると思う。

いかもしれない。
反発係数が50パーセントで頭打ちになっているせいで，跳ね方はあまり派手ではない。さすがに何回も見ているうちに飽きてくると思うが，フレームファイルができるのにそれほど時間がかからないので，落ち始めの高さとか視点の位置とかいろいろ変えてみて，気にいったものが決まればRENDにかけて，綺麗な画像を堪能するのが通の遊び方だ。

球の落下をシミュレートする

さて，ball( )という関数でフレームファイルを生成していると前述したけど，このなかでやっていることというのは，シミュレートする現象が単純なだけあってやはりたいしたことはない。ただいくつかポイントがあるので，その部分だけ説明をしよう。
まず視界の中心には，常に球体があるようになっている。つまり球を追いかけるように見る向きを変えているのだ。と書くとなにか難しそうだけど，実際はそうでもない。というのは，DōGAのフレームファイルでは視点のほかに注目点というのを指定するようになっており，常に注目点が画面中央にあるように，画像が生成される。つまり，球の座標を注目点として設定すれば，勝手に球体を追いかけてくれるのだ。
ところで，高校の物理で，

$$y = v_0t + 1/2gt^2$$

という公式を習う。いわずと知れた自由落下の公式だ。まずこの公式で球の高度を求めて，同じ座標で注目点と球を書き出す。たったこれだけのことなのだ。
最初にプログラムを打ち込んでからこの文章を読む人はいないと思うが，ball()で吐き出された画像を見てみると，床に落ちた影が，リアルさをかもし出すのに大きな役を担っていることに気づくはずだ。この影を落とすということも，意外と簡単にできてしまう。
ご存じのとおり，レイトレーシングでは物体と反射率，光源などを設定するだけできちんと影を落としてくれる。その代わりもの凄く時間がかかる。一方DōGAはどうかというと，速いには速いのだ。ただし，シェーディングと呼ばれる，物体への光の当たり方の差によって生じる物体自体の影は表現してくれるけど，物体が相互におよぼす光の現象まではシミュレートしてくれない。だから，速さの代償として切り捨てられた影を，人間の英知で取り戻さなければならないのだ。

地面などに落ちる影のカタチを一般的に定義すると，ある物体の輪郭を影の落ちる対象面に平面投射したカタチ，となるだろうか。さて，ここで嬉しいことに，いま影を落とそうと思っている球体というのは，どのような面に投射しても，同じ半径の円になるのである。球体を真っ二つに輪切りにした切り口が，ちょうどその円に当たる。鋭い人はとっくに気づいていると思うが，要するに球の影というのは，平坦な面に落ちる場合はいつも，同じ半径を持つ円になるのである。つまり影として用意する形状はひとつでいい，ということなのだ。このために，僕はあらかじめ“shadow.SUF”を用意しておいてくれといったのだ。
これで残された問題は，影はどこに落ちるか，ということだけとなった。これもたいして難しくはなく，光の差す方向を考えればすぐわかる。影というのは，光源と物体を通る線と，影の落ちる面との接点に落ちるのである。ただしDōGAの場合，光の方向だけを設定するようになっている。つまり光源は無限大の彼方にあるので，この場合光のベクトルだけを考えればいい。
3260行で光のベクトルを設定している。これを長さ1のベクトルに変換して，X，Y成分と，球の高さ，すなわちZ座標とをそれぞれかけ合わせることで，影の落ちる座標が決定する。1430行でその計算をやっている。数学的には高一レベルだ。簡単でしょ？
ところで，この影のアトリビュートの透明度を50パーセントくらいに設定すると，これはなんとお得なことに，乱反射の影になる。あと，影の座標を光の差し込む方向に少しずらして，球の中心の，地面に対する影の投射点を中心に，120度ずつ回転させた影を3つ用意すると，スポット光源3個の影ができあがる（ややこしい表現でごめん。日本語だから，ある程度しようがないのだ)。ここまですると，もうレイトレレベルの影である。
さて，最後になるけど，少々厄介な問題を取り上げなければならない。リストでいうと，1470～1530行の部分だ。球体が地面に当たって，跳ね返るときの処理である。え，なにか難しいことがあるのかって？僕も最初はそう思っていたのだ。でもってさらりと流したら，痛い目を見た。
球体が地面に当たったかどうかは簡単に計算できる。また，そのときの落下速度を計算して，反発係数をかけたものが跳ね返りの初速となる。また方向が上向きなので，初速はマイナスになるだろうか。

これがさらりと流したかたちだ。ball( )という関数の衝突判定部分の初期設計はまさにこのとおりであった。しかしこれをいざ走らせてみると，ボールがいつまでも跳ねていて，なんと止まらないのである！
でいろいろと検討してみた結果，原因が判明した。ここから話がかなり込み入ってくるので，心して読んでほしい。
まず，落下に関係があるのは，落ち始める高さと，初速と，あと落ち始めてからの時間である。話を単純化するために，自由落下の場合に限定しよう。これで初速は考慮しなくていい。また，落ち始める高さも定数項なので考えないことにする。
さて，残ったのは時間である。ちなみにDōGAでは，20分の1秒をひとコマとして扱うので，ここでは変数bを1フレームごとに1増して，先の公式に当てはめるときは20で割って対処している。つまり，時間の変数bは常に整数である，ということだ。
ところが，球が地面に激突する瞬間というのは，必ずフレームとフレームの境目にあるとは限らないのである。たとえば，20コマ目に地面と衝突するという計算結果が出たとしようか。しかしbは整数範囲でしか増加しないので，19.5コマ目のことなどは考慮に入っていない。つまりある時点で球の位置が地面より低くなったという判定が下ったとしても，それはそのときにはすでに地面にめり込んでいる，ということを意味しているにすぎない。
そこで，まず球が地面にめり込んでいることがわかった時点で，厳密にいつ，地面と衝突するかを計算する。その結果算出された時間をもとに，初速を割り出す。
そしてここからが肝心なのだが，フレームとフレームの途中で球が地面と衝突するということは，跳ね返った時点から，20分の1秒たたないうちに，つまりbが1増えないうちに，次のフレームがくるのである。もっとわかりやすくいうと，コマとコマの途中で球が地面と衝突するのだから，その時点を，時間ゼロとしなければならないのだ。
どうやらこのときに出る計算誤差が跳ね返りの初速にたまって，それで球体がいつまでも跳ねて止まらないといったことが起こったらしい。解決策として，先に求めた衝突の時間の小数部分を取り出して，それを1から引いたものを，時間のカウンタbに代入している。また，それにともなって落ち始める高さも設定する必要があり，その2つの処理を1510，1520行で行っている。
あとはこれらの作業を，跳ね返りの初速

が十分に小さくなるまで機械的に繰り返すだけである。

実をいうと，この落下する球体というのは，初めは作る予定のなかったものなのだ。いきなり難しいものに取り組むのは心もとなかったので，試しに作ってみたにすぎない。今回の目玉は，なんといっても次に説明する落下し，しかも跳ね回る花瓶をシミュレートしたvase( )という関数にあるのだ。

跳ねる花瓶に苦悶する

いくらCGとはいえ，いくら自分のマシン上で走るからといえ，自由落下する球体を何回も見ていれば，さすがに飽きがくる。なぜ飽きてしまうかというと，動作を規定する因子があまりにも少なすぎて，どのように動くのかが簡単に予測できてしまうからである。何回も見ているうちに，だんだんと，

「ケッ，どうせ跳ねるだけだぜ」

という思いが頭をもたげてくる。だから冒頭にあんなでかいことをいった僕は，これだけで今回の仕事を終わらせるわけにはいかないのだ。

で，落下球体ができあがったので今度は花瓶だとばかりに手をつけると，これが，思ったよりはるかに難しいのだ。

プログラムを組むときというのは，誰でもある程度の指針みたいなものを持って臨むのだと思うが，僕も，

「ココがあーなるからこー，でアソコはこんなん」

と，一応プログラムの初期設計を頭の中ですませてから，画面に向かった。

ところがだ，初期設計どおりにプログラムを組んでみて，出てきたフレームファイルの動作確認をしてみると，これがもの凄く不自然なのだ。

花瓶であるから，地面との衝突判定は球体の部分と口の部分の2カ所で行う。この部分は先の球体でレクチャーずみだったので，花瓶が地面にめり込んだりすることはないし，また跳ね返ることは跳ね返るのだ。ではなにが不自然かというと，花瓶は球のように，地面と衝突する点と重心とを結ぶ点が，常に地面に対して垂直ではなく，したがって横っ飛びしたり，跳ね返ってくるりと回ったりする。この動きが，どうもうまく再現できないのである。

そこで僕は，論より証拠と諺にならって，実験をしてみることにした。家中探し回ったら，ちょうど丸底フラスコのような形をした，化粧品のビンが見つかった。こいつを何回も落としてみて，どんな振る舞いをするか研究するのだ。

最初は自分の部屋で落下実験を行っていた。だけど僕の部屋は畳敷きであり，ビンの跳ね返りがよくない。そこでふすまをガラリと開けて，板敷きの廊下で，ニュートンよろしく物理実験に勤しんでいた。口から落ちるときはどうなるかとか，落とすときに回転を与えるとどうかとか，いろいろ試しているうちに薄ボンヤリと，問題の核心が見え始めた。

高校物理を思い出す

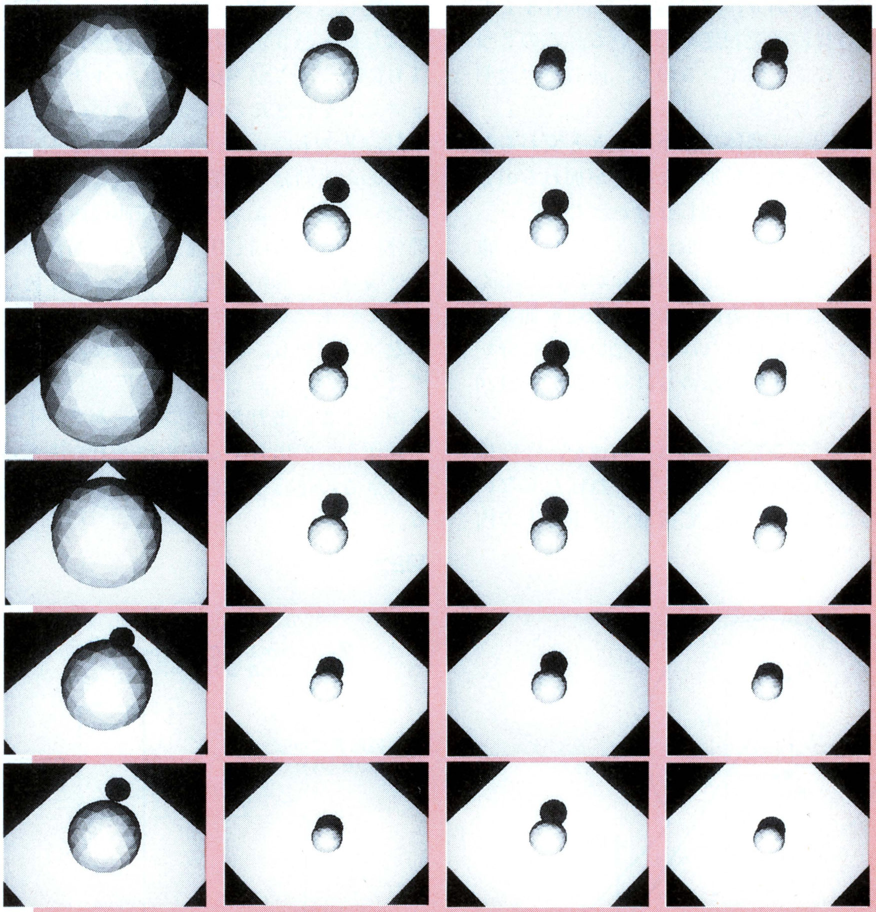
まず，この花瓶の落下の場合，問題は3つに分かれる。ひとつ目が跳ね返りの問題。2つ目は跳ね返り時の横飛びの問題。そして最後は跳ね返りによって生じる花瓶の回転の問題である。順を追って説明していこう。

まず第一の問題と第二の問題は大きな関わりを持っている。というのは，前者は跳ね返りのベクトルの縦成分だし，後者は横成分だからだ。図1を見てほしい。反発力Rは，地面と花瓶との接触点と重心を通る線上にある。だから接触点と重心との内積をとれば，この2つの成分は簡単に抽出できる。これだけで横飛びの問題は解決される。

跳ね返りの縦成分については，もうひとつ，重力を考慮に入れなければならない。図でいうとGの矢印がそれにあたる。見てわかると思うが，GとRyは互いに逆向きであり，この2つを足して初めて，最終的に重心にかかる力が導出される。

さて，最後に残ったのは衝突によって生じる回転の問題だ。ちなみに，図1の状態で地面に落ちると，左回りに回転を始めるであろうことは推測がつく。

で，いったいどのような原理で，この回転が生じるのだろうか。物理はあんまり得意でなかったのではっきりしたことはわからないので，いくつか試してみてもっとも本物らしく見える理論を採用するという方



上の写真はball( )部分の出力結果をレンダリングして連続撮影したもの。必ずしも等間隔ではないので，パラパラアニメには向いていない。なお，ここで使用している球体(？)はTAMEN.Xを用いて作成されたもの。あえてスムースシェーディングは行わず，半透明にしてみた。ちなみに95ページの花瓶は回転体で作られている。

法を僕はとった。そんな風にひねり出したのが，以下のような理論だ。

また図1を見ていただきたい。跳ね返る瞬間，重心には重力と反発力の2つの力が働き，結果として反発力は減衰される，と前述した。で，地面との接触点ではどうかというと，これは重力の減衰は受けずに，サラのままのRyがかかるらしい。

つまり，重心の動きと接触点の動きのあいだに差が生まれるようで，この差分だけ，花瓶は回転しようとするらしい。図2を見るとわかりやすいと思うけど，重心との距離と，いまいった動きの差とを使って内積をとり，それをATAN（逆三角関数！）で角度に変換するとうまくいく。

と，ここまで理論的な後ろ楯が完成すれば，あとはこれをプログラムとして書きあげるだけである。順番が多少前後するかたちになるけど，まずは衝突判定部から説明しよう。

繰り返しになるが，衝突判定は2つに分かれている。2000～2280行までの部分で花瓶の丸底の部分，その後から2550行までで花瓶の口の衝突判定をしている。この2つは互いによく似ているので，前者だけ説明することにする。

原理的には球体の落下のときと変わらないが，いくつか付加的な要素が加わるのでそいつをうまく処理しなければならない。まず花瓶の衝突後の飛び出し速度には，重力と，もうひとつ花瓶の回転が関係してくる。また，回転方向によって初速を加算する場合と減じなければならない場合に分かれる。その初速を求める計算を2050～2120行で行っている。

さて，ここで求められた初速を，今度は縦，横の各成分に分解しなければならない。そのために，2150行で接触点と重心との内積（図1参照）を求め，さらに次の行でその内積をSINに変換している。ここまでが下準備で，これらを成分分解したり公式に当てはめたりして，実用の変数に割り当てていく。

まず，図2の法則に基づいて，花瓶の回転角の変化分の内積を求める。それをTANに変換してATANにかけ，角度を導き出し，回転角の増分である変数hに干渉させる。だっ，誰だっ，そこでサジ投げてるのは！

2200，2210行で，反発力の横成分をさらに3次元座標上のX，Y成分に分解して，横飛びに用いる変数に足している。リスト上では引くかたちとなっているが，これは便宜上こうなっているだけで，かたちとしては足しているのだ。

と，ここまでくればあとは球体の落下のときと同じように，半端分の時間と高さを求め，跳ね返りの縦成分を抽出して初速の変数Cにぶち込めば，衝突判定はおしまい。ふうっ。

今回の僕の苦労のほとんどは，この花瓶の衝突判定に込められているといって過言ではないと思う。これだけのために，延々4日間あまりも悩み続けた。マルチステートメントをほとんど使っていないのに，どうだ，リストのこの込み入りようは！

ええ，余談はさて置き，最後に花瓶の影について説明しよう。落下球体の説明の折，球体の影は常に同じ円となるので都合がいいと書いた。しかし花瓶の場合，回転角によっていくつか違う影を用意しなければならない。だから，影を落とすのはムリ……ではないのだ。あとでもう一度詳しく触れるが，ここで出力されるフレームファイルを画像として出力するために，ある決まった大きさの花瓶を用意してもらうことになる。

で，その花瓶というのは丸底部分は完全な球体で，底から口が飛び出している形になる。要するに問題は，このひょこっと飛び出している口の部分である。これさえなんとかすれば，あくまでも疑似的にではあるけど，影を落とすことができる。

まず，丸底部分はそのまんま，球体のときに使った円を影とする。そしてそれと部分的に重なるように，首の部分の影を書き出すのである。ただし，花瓶の回転角によって突き出方がさまざまであるので，それはまたヤヤコシイ計算で対処しなければならない。

1930～1950行で，その口の部分の影が，どこに落ちるかという計算をしている。花瓶の口の先端部だけを見て，その座標をもとにどこに影が落ちるかを計算し，座標を書き出している。また回転角はZ軸周りだけを考えればよく，花瓶の回転角がそのまま使える。プログラムの説明はこのくらいだろうか。

出てきた画像にカンドーする

なんだかんだと長いリストだけど，原理なんかわからなくていいから，とにかく打ち込んでみていただきたい。出てきたフレームファイルを動かしてみれば，どうして僕がこんなに苦労したかがわかってもらえると思う。

で，その前に，球体のときと同じく，あらかじめ用意しなければならない形状ファイルが2つある。この関数で出力されるフレームでは4つの形状を使っているけど，そのうち2つ，床と花瓶の丸底の部分の影は球体のときに作ったものを流用するので十分だ。

そのほかに用意してもらうもののひとつ目は，まず主役の花瓶である。以下の条件を満たしていれば，派手な色をしていようが，表面に，いかにもファットメイジが飛び出てきそうなマッピングがしてあろうが構わない。取っ手なんかを付けられては困るけど。

まず丸底部分は，中心をZ軸上，Z＝－20のところに取った真球であること。また，Z軸に沿って，つまり上方向に口が突き出ていて，その高さはZ＝80で，口の広さ，つまり口の円の半径は15であること。

これも球体を作るときと同じように，まず(0,0,－20)を中心にダミーの円を作っておいて，消したら次に回転体作成を指定し，指定どおりの口を切り出して，続けて半円を描いて面確定というパターンがいいと思

図1

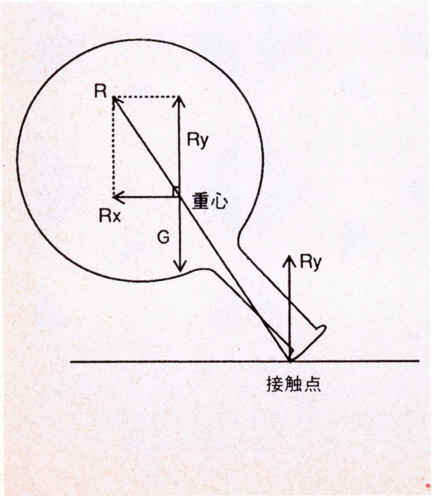
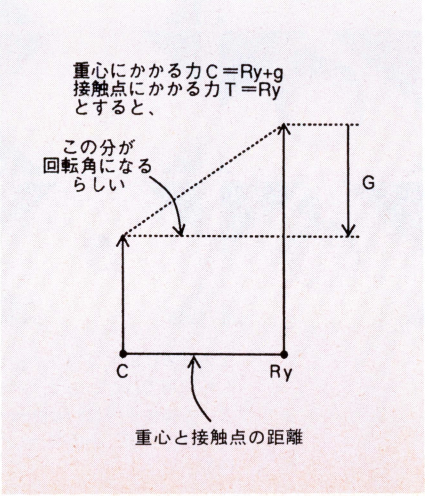


図2



う。要するに，球の部分と口の部分を分けないで，一気に回転体で書きあげてしまえばいいのだ。TAMEN.Xで作った綺麗な球体を加工してもいいが，首との接続を滑らかにするには回転体を使ったほうがラクである。

さて，もうひとつ用意してもらう形状というのは，わかるとは思うが花瓶の口の影に当たる部分である。こちらの条件は，花瓶を縦に2等分したときの口の形と同じで，XY平面上にあるもの。面倒臭かったらただの四角形で構わない。

(15,80,0)　　(-15,80,0)

(-10,30,0)　　(10,30,0)

の4点を結べば，お望みの図形ができあがる。

走らせてみて気づくと思うが，落下球体のときと違うのは，初めに入力する変数の種類が増えているということだ。回転角が2軸に，その上Y軸の回転角の増分まで指定できてしまう。つまり，落ちながら回転する部分までシミュレートしているのだ。うーん，我ながらいい仕事をするぜ。

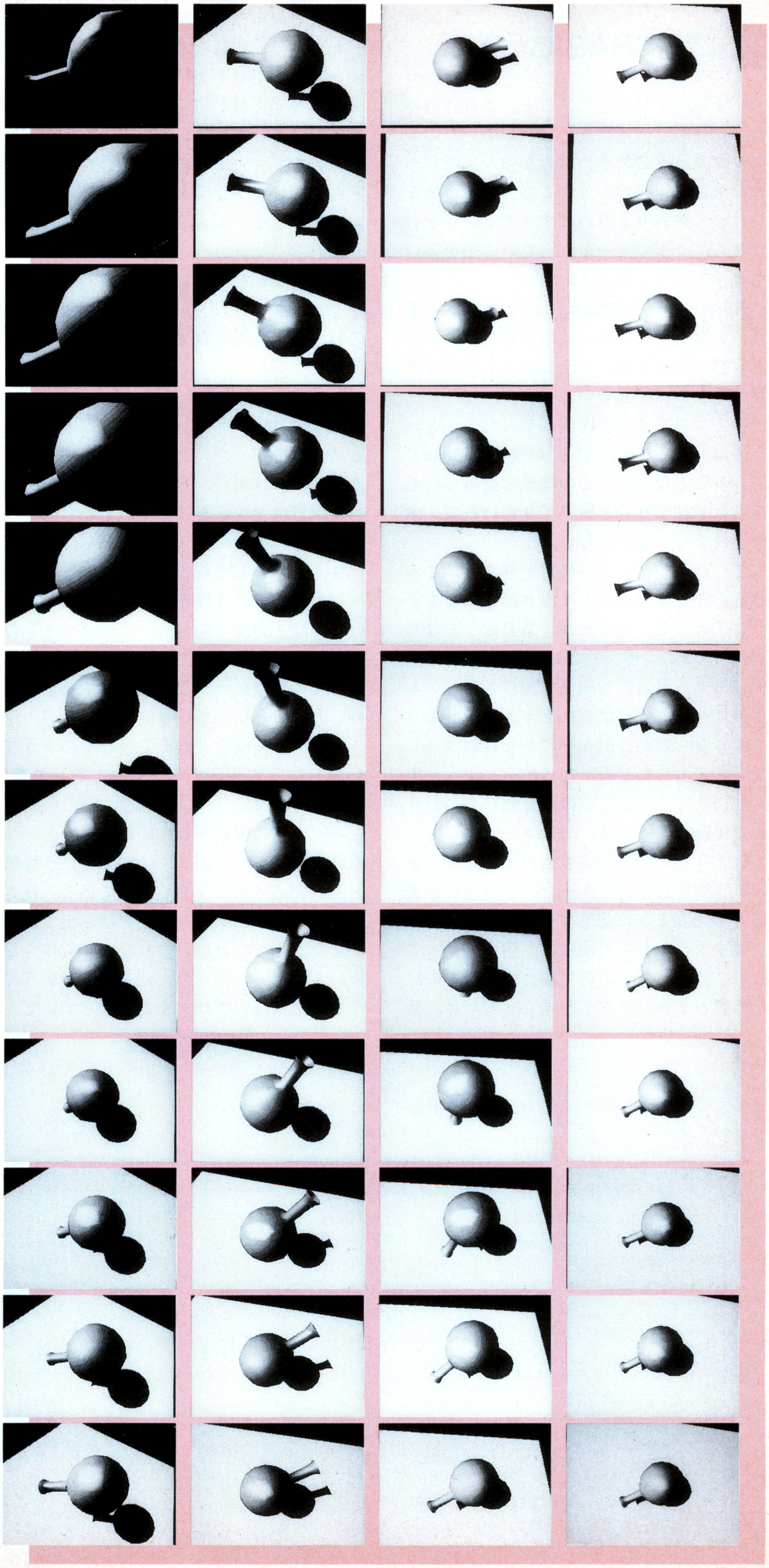
なお，僕のおすすめは，落ち始める高さに500，Y軸の回転角に70，その増分に4というやつだ。残りの変数は適当に設定してもらって，とにかくフレームファイルを吐き出して，画像として出力してごらん。けっこう，くるものがあると思うぞ。

バグ取りのため，いろいろな設定で花瓶を落としてみたけれど，これが意外にガンバルのだ。丸底部分と花瓶の口がちょうど同時に床と接触するようにとか，かなりいやらしく設定してみても，ちゃんと本物らしい絵を出力する。

手前味噌で申しわけないけど，ここまで本物らしいと，僕はもう神様にでもなったような気になる。ああっ，これだ，これだったんだ，僕の探していたものは。ここ数日というもの，なんともいえない充実感に，僕は包まれている。

ところで，ここで使った方法論を発展させれば，まだまだいろんなことができそうだ。たとえば，板が粉々に割れ，地面に当たってちりぢりに跳ね飛ぶなんてこともできるだろう。要するに，vase( )で使った衝突判定部を一般的な形に書きなおして，物体の角頂点に対してそれぞれ衝突判定を行えばいいのだ。

あと，この判定部分を人体モデルの四肢に適用して，体操競技真っ青のアクロバットをやらせてみるとか……，あっ，先を越されるとヤなのでこれくらいにしとこうっと。



どうしてBASICか?

いま原稿を見返してみると，これはいったいプログラムの特集のために書いたのか，グラフィックの特集のためなのか判断がつかないほど，内容がてんこ盛りの感がある。だけど「終わりよければすべてよし」というから，ここでもっともらしいことを書いてゴマかしてしまおう。

X68000という機械は，かなり凄いと思う。いきなりなにをいい出すんだコイツはとお思いだろうが，もうしばらく僕のヨタ話につきあってもらいたい。

考えてみると，今回僕がやったことというのは，(当たり前だけど)標準のX68000でできることなのだ。この機械を持っていて，あとは本誌7月号を買ってさえいれば，余分な投資は一切いらない。

で，ほかのパソコンに目を移すと，この状況の優越性はいっそう浮き彫りになるように思われる。たとえば，某P機などは問題外だ。また，機能的にはX68000と近い関係にあり，最近教育関係で注目されているT機についても，プログラミングをしようとするなら新たに財布を開かなければならない。どちらにしろ，なにかをしようとすると決まって安からぬ投資を強いられたり，また場合によっては，一般ユーザーにとってはまったく門戸が閉ざされていたり，その販売姿勢自体が，ユーザーの創造心を裏切っているように見受けられるのだ。

要するにこれらの機械は，体よくいってせいぜい，ソフトの再生機にすぎないのだ。ただ買ってきたソフトを走らせるだけ，情報を受け取るだけで吐き出すことをしない。しかもその情報は次々生み出されるので，追いかけるだけでも精一杯で，つまりいつも適当な充実感が得られるような構造ができあがっている。猿にマスターベーションを教えると一生やってるというが，まさにそれと同じような図がここでは展開されており，そしてこちらのほうが，むしろより一般的なパソコンライフなのである。

この文章が載る頃にはもう終わっているかもしれないけれど，このあいだ国営放送のパソコン入門を見ていたら，先生とおぼしきオヤジが，

「ソフトは，もう作る時代ではなくなりました」

とヌカしやがった。ケッ，嘘をいっちゃ困るぜ。それは単に，P機のような奇形のパソコンではソフトを作る気がしないという話じゃないのか？

よくよく考えてみると，X68000という機械が形成している世界の，その住人たちというのは，異常な創造意欲のカタマリみたいなやつらだ。十数万という出荷台数からはとても想像のつかないような量の情報を，外に向けて発信している。

そしてその状況を温かく包むかのように，Cがあり，アセンブラがあり，そしてX-BASICがある。

ユーザーの側が，なにかをしようという意欲を持っている限り，X68000という機械とその周辺は限りなく開かれている。残された問題は，ユーザーが足を踏み出すか否かということだけだ。作れないから作らないのではない。作りたいから作れるようになるのだ。このプログラムを作っていて，僕が痛感したことである。

もっといおう。僕は実は，文系の学部に通う，どうってことない大学生なのだ。つまり，数学とかCGについて，特別な教育を受けたわけではない。僕の持っているその種の知識というのは，せいぜい高校レベルで，バリバリの理系学生のアナタなんかには到底かなうはずはない人なのだ。

そんな僕にさえ，これくらいのことはできてしまうのである。だったら，あなたたちにできないという法はどこにもないはずだ。短絡的すぎるって？　だけど，そういうことじゃないか？

＊　＊　＊

ところで，僕が実験に使った化粧品のビンなのだけど，調子に乗ってかなり高いところから，廊下のフローリングに向けて落としたもんだから，とうとう割れてしまったのだ。その化粧品は母親のもので，壊したのが見つかると大目玉である。僕は破片を急いで片づけて，ゴミは母の目に触れないように，外のダストボックスに捨てにいった。

ところが，廊下を拭いた雑巾に，その化粧品の匂いが残っていたらしく，母親はそれをめざとく見つけて，

「こんなことするの，アンタしかいないわよね」

「だいたいアンタは小さいときから」

と，あることないこといい出したので，正直にことの原因を話して聞かせた。するとバイトで壊したんだから，原稿料で弁償しなさいという。値段を聞いてみると，これが法外に高いのだ。女ってどうして……。ううっ，お母ちゃんのバカッ！

リスト1

```
1000 /*
1010 /*  BASICでDoGAのフレームファイルをはき出す
1020 /*
1030 /*                              プログラムを作った人：俺
1040 /*                                            原案：僕
1050 /*                          Special thanks to DoGA
1060 int i : float x1,y1,z1,ex,ey,ez,rz,ry,rx,a,b,c,d,e,f
1070 float g=980.665#,h,j,k,l,m,fmax,x,y,z,dx,dy : str fn,st
1080 screen 2,0,1,1
1090 print "どちらを御所望ですか?"
1100 print "落下する球体                  (1)"
1110 print "落下し、しかも跳ね回る花瓶 (2)";
1120 st=inkey$
1130 if st="1" then {
1140 ball() } else {
1150 vase() }
1160 mktch(int(f))
1170 print "作業を終了しました。"
1180 end
1190 /* ///////////////////////////////
1200 /* ////////// 落ちる球体 ///////////
1210 /* ///////////////////////////////
1220 func ball()
1230   cls : a=0 : b=0
1240 print "落下する球体"
1250   repeat
1260     asv()              : /* 各変数の設定
1270     fn=fnin()          : /* ファイルネームの入力
1280   until yorn()=0
1290   print "フレームファイルを生成します。準備は";
1300   yorn()
1310   e=csrlin : f=1 : b=0 : c=0
1320                  /* ちなみに、a は落ち始める高さ、
1330                  /* b は時間、c は跳ね返りに用いる初速
1340   fopen(fn+".frm","c")
1350   repeat
1360     locate 0,e : print "フレーム :";f
1370     top(f)          :  /* フレームのヘッダ
1380     ep(ex,ey,ez)    :  /* 視点の座標
1390     z1=grav(a,b,c)  :  /* 座標計算
1400     ap(0,0,z1)      :  /* 注目点
1410     obj(0,0,z1,0,0,0,"ball")
1420     obj(0,0,0,0,0,0,"floor")
1430     x1=z1*-0.557# : y1=z1*-0.3714# : /* 影の座標計算
1440     obj(x1,y1,1,0,0,0,"shadow")
1450     b=b+1 : f=f+1
1460     print using "球の高さ ###.##";z1
1470     if (grav(a,b,c)-50)<=0 then {
1480        b=(-c+sqr(pow(c,2)+(2*g*(a-50))))/g*20
1490                                          /* ↑これが大ミソ！
1500        c=-(c+g*(b/20))*(d/100)       : /* 跳ね返りの初速
1510        b=1-(b-int(b))                : /* ←ここは中ミソ
1520        a=grav(0,b,c)+50              : /* ←ここは小ミソ位...
1530     }
1540     tale()           :  /* フレームを閉じる
1550     if c<>0 and c>-20 then c=999
1560     if f=fmax then c=999
1570   until c=999        :  /* 跳ね返りが十分小さくなるまで
1580   fclose(0)
1590 endfunc
1600 /* ///////////////////////////////
1610 /* ////////// 落ちる花瓶 ///////////
1620 /* ///////////////////////////////
1630 func vase()
```

```
1640   cls : a=0 : b=0
1650 print "落下し、しかも跳ね回る花瓶"
1660   repeat
1670     asv()              :  /* 各変数の設定
1680     fn=fnin()          :  /* ファイルネームの入力
1690     input "Z軸の回転角の初期値",rz
1700     input "Y軸の回転角の初期値",ry
1710     input "Y軸の回転角の増分",h
1720   until yorn()=0
1730   print "フレームファイルを生成します。準備は";
1740   yorn()
1750   e=csrlin : f=1 : b=0 : c=0
1760                   /* ちなみに、a は落ち始める高さ、
1770                   /* b は時間、c は跳ね返りに用いる初速
1780                   /* h はX軸の回転角の増分
1790                   /* dx,dy は x,y 座標の増分
1800   fopen(fn+".frm","c")
1810   repeat
1820     locate 0,e : print "フレーム :";f
1830     top(f)           :  /* フレームのヘッダ
1840     ep(ex,ey,ez)     :  /* 視点の座標
1850     z1=grav(a,b,c)   :  /* 座標計算
1860     ap(x1,y1,z1)     :  /* 注目点
1870     obj(x1,y1,z1,rz,ry,0,"vase")
1880     obj(0,0,0,0,0,0,"floor")
1890     j=sin(rad(ry))*20 : k=z1-cos(rad(ry))
1900     x=x1-k*0.557#-cos(rad(rz))*j
1910     y=y1-k*0.3714#-sin(rad(rz))*j
1920     obj(x,y,1,0,0,0,"shadow")            : /* 花瓶の球の部分の影
1930     j=sin(rad(ry))*80 : l=sgn(j)
1940     x=x1-z1*0.557#+cos(rad(rz))*(abs(j)-80)*l
1950     y=y1-z1*0.3714#+sin(rad(rz))*(abs(j)-80)*l
1960     obj(x,y,1,rz-90*l,0,0,"shadow2")    : /* 花瓶の口の影
1970     b=b+1 : f=f+1 : ry=ry+h : i=0
1980     x1=x1+dx : y1=y1+dy           : /* これがあるから横っ飛びする
1990     print using "花瓶の高さ ###.##  回転角 ###.##";z1,ry
2000  /* @@@  まず丸底部分が地面に触れたかどうか調べる  @@@
2010  /*
2020     y=cos(rad(ry))*20+50              : /* 丸底の中心の高度
2030     x=sin(rad(ry))*15*sgn(-sin(rad(ry)))+cos(rad(ry))*80
2040     l=grav(a,b,c)-y
2050     if l<=0 and l<grav(a,b,c)+x then {
2060       if l<=0 and (a<y) then {
2070       b=(-c+sqr(pow(c,2)+(2*g*(y-a))))/g*20
2080       } else {
2090       b=(-c+sqr(pow(c,2)+(2*g*(a-y))))/g*20 }
2100       k=sin(rad(h))*400*sgn(sin(rad(ry)))
2110                                   /*  回転による落下速度の増加
2120       c=-((c+g*(b/20))-k)*(d/100)     : /* 跳ね返りの初速
2130  /* ここら辺から結構ややこしい....
2140       x=sin(rad(ry))*20
2150       j=x/sqr(pow(x,2)+pow(y,2))      : /* (x,y)の内積を求める
2160       k=sqr(1-pow(j,2))               : /*  cosからsinを求める
2170       l=sqr(pow(x,2)+pow(y,2)):l=l/sqr(pow(l,2)+pow(c/20,2))
2180       l=atan(sqr(1-pow(l,2))/l)*180/pi()*sgn(x)
2190       h=h+l                           : /* 回転角の増分に干渉させる
2200       dx=dx-cos(rad(rz))*c*j/40*sgn(x) : /* 跳ね返りの x 成分
2210       dy=dy-sin(rad(rz))*c*j/40*sgn(x) : /* 跳ね返りの y 成分
2220       if f-m<2 then {
2230            dx=0 : dy=0 }
2240       m=f
2250       b=1-(b-int(b))
2260       a=grav(0,b,c)+y
2270       c=(2*c*k)-c : i=2                : /* 跳ね返りの Z 成分
2280     }
2290  /* @@@  次に花瓶の口が地面に触れたかどうか調べる  @@@
2300  /*
2310     y=sin(rad(ry))*15*sgn(-sin(rad(ry)))+cos(rad(ry))*80
2320                                       /* 花瓶の口の高度
2330     if (grav(a,b,c)+y)<=0 then {
2340       if (grav(a,b,c)+y)<=0 and (a+y<0) then {
2350       b=(-c+sqr(pow(c,2)+(2*g*(-a-y))))/g*20
2360       } else {
2370       b=(-c+sqr(pow(c,2)+(2*g*(a+y))))/g*20 }
2380       k=sin(rad(h))*1600*sgn(sin(rad(ry)))
2390                                   /* 回転による落下速度の増加
2400       c=-(c+g*(b/20)+k)*(d/100)    : /* 跳ね返りの初速
2410       x=cos(rad(ry))*15*sgn(-sin(rad(ry)))-sin(rad(ry))*80
2420       j=x/sqr(pow(x,2)+pow(y,2))      : /* (x,y)の内積を求める
2430       k=sqr(1-pow(j,2))               : /*  cosからsinを求める
2440       l=sqr(pow(x,2)+pow(y,2)):l=l/sqr(pow(l,2)+pow(c/20,2))
2450       l=atan(sqr(1-pow(l,2))/l)*180/pi()*sgn(x)
2460       h=h+l                           : /* 回転角の増分に干渉させる
2470       dx=dx+cos(rad(rz))*c*j/40*sgn(x) : /* 跳ね返りの x 成分
2480       dy=dy+sin(rad(rz))*c*j/40*sgn(x) : /* 跳ね返りの y 成分
2490       if f-m<2 then {
2500            dx=0 : dy=0 }
2510       m=f
2520       b=1-(b-int(b))
2530       a=grav(0,b,c)-y
2540       c=(2*c*abs(k))-c : i=i+1         : /* 跳ね返りの Z 成分
2550     }
2560     tale()          :  /* フレームを閉じる
2570     x=cos(rad(ry))*20+50
2580     if c<>0 and int(abs(x+y))=0 and abs(c)<5 then c=999
2590     if f=fmax then c=999
2600   until c=999
2610   fclose(0)
2620 endfunc
2630 func asv()          :  /*@@        各変数の設定
2640     input "落ち始める高さ",a
2650       if a<100 then a=100
2660     input "反発係数 ( 0 - 50% )",d
2670       if d>50 then d=50
2680       if d<0 then d=0
2690     input "フレーム数の上限",fmax
2700       if fmax=0 then fmax=10000
2710     input "視点の位置  x:",ex
2720     input "            y:",ey
2730     input "            z:",ez
2740 endfunc
2750 func str fnin()       :  /*@@ファイルネームを入力する
2760  int a,i : str h
2770   repeat
2780     input "フレームファイルの名前",h : i=1
2790       if instr(1,h,".") then {
2800         print "拡張子は不用です" : i=0}
2810   until i=1
2820   return(h)
2830 endfunc
2840 func int yorn()      :  /*@@      いいか悪いか聞く
2850  str a
2860   print "よろしいですか?";
2870   a=inkey$ : print
2880   if a="y" or a="Y" then {
2890   return(0)  } else {
2900   return(-1) }
2910 endfunc
2920 func float grav(a;float,b;float,c;float)
2930  float d             :  /*@@ 重力を加味した座標計算
2940   d=a-(c*(b/20)+0.5#*g*pow((b/20),2))
2950   return(d)
2960 endfunc
2970 func mktch(f;int)    :  /*@@  タイムチャートを作る
2980   print "タイムチャートを作成します"
2990   fopen(fn+".tch","c")
3000  str a,b
3010                a=".timechart" : wrline(a)
3020                a=".wait 10     "+fn+"001"        : wrline(a)
3030   b=plz(f-2) : a=".wait 1      "+fn+"[2-"+b+"]" : wrline(a)
3040   b=plz(f-1) : a=".wait 10     "+fn+b           : wrline(a)
3050   fclose(0)
3060 endfunc
3070 func str plz(a;int)
3080  str b
3090   b=string$(3-len(str$(a)),"0")+str$(a)
3100   return(b)
3110 endfunc
3120 func float rad(r;float)
3130   return(r/180*pi())
3140 endfunc
3150 /* ////////////////////////////////
3160 /* //////     ファイル入出力関係の関数    //////
3170 /* ////////////////////////////////
3180 func top(fm;int)     :  /*@@ フレームの共通ヘッダを @@
3190  str e[64]           :  /*@@  ディスクに書き出す   @@
3200   e="/********** frame "+right$(str$(fm),3)+" *********/"
3210   wrline(e)
3220   e="fram" : wrline(e)
3230                             /*フレームの始まり
3240   e="{"   : wrline(e)
3250   e="light pal( rgb ( 1.00 1.00 1.00 ) "
3260   e=e+"-3.00 -2.00 -4.00 )" : wrlint(e)
3270                             /*光源のパレットと方向
3280 endfunc
3290 func ep(x;float,y;float,z;float)
3300                             /*@@      視点の座標を指定
3310  str a,b,c,e[64]
3320   a=pl(x) : b=pl(y) : c=pl(z)
3330   e="{ mov ( "+a+" "+b+" "+c+" )  eye  deg( 60 )"
3340   wrlint(e)
3350   e="}" : wrlint(e)
3360 endfunc
3370 func ap(x;float,y;float,z;float)
3380                             /*@@        注目点を指定
3390  str a,b,c,e[64]
3400   a=pl(x) : b=pl(y) : c=pl(z)
3410   e="{ mov ( "+a+" "+b+" "+c+" )  target"
3420   wrlint(e)
3430   e="}" : wrlint(e)
3440 endfunc
3450 func obj(x;float,y;float,z;float,r1;float,r2;float,r3;float,
nm;str)
3460  str a,b,c,e[128]    :  /*@@      物体名nmを書き出す
3470   a=pl(x) : b=pl(y) : c=pl(z)
3480   e="{ mov ( "+a+" "+b+" "+c+" )  "
3490   a=pl(r1) : b=pl(r2) : c=pl(r3)
3500   e=e+"rotz( "+a+" )  roty( "+b+" )  rotx( "+c+" )  obj "+nm
3510   wrlint(e)
3520   e="}" : wrlint(e)
3530 endfunc
3540 func tale()          :  /*@@    フレームを閉じる
3550   fwrites("}"+chr$(13)+chr$(10),0)
3560 endfunc
3570 func str pl(f;float) : /*@@      入力された数字の
3580  str b[10]           : /*@@       桁を揃えて返す
3590   b=str$(int(f*100)/100)
3600   if len(b)<6 then b=space$(6-len(b))+b
3610   return(b)
3620 endfunc
3630 func wrline(a;str)   :  /*@@  文字列と改行コードを
3640                         /*@@   ディスクに書き出す
3650   fwrites(a+chr$(13)+chr$(10),0)
3660 endfunc
3670 func wrlint(a;str)   :  /*@@      タブ文字を添えて
3680                         /*@@        一行書き出す
3690   fwrites(chr$(9)+a+chr$(13)+chr$(10),0)
3700 endfunc
```

オブジェクト指向に学ぶ

作り散らかせます

Tan Akihiko 丹 明彦

Programing

大きなプログラムを作るのは困難です。それは誰でも同じ。ならばどうすれば楽になるか……。という問題の解答のひとつがオブジェクト指向によるアプローチです。これをSX-WINDOWプログラムへ応用してみましょう。

大作プログラムを作り上げるには

僕は基本的に小物プログラムを作るのが好きだ。ちょっとした画像処理プログラムやテキストファイルを整形するフィルタプログラムなんかは大好物。逆に，作業場が狭いのは嫌い。その点，X68000は申し分ない。広い開発環境で小さなプログラムを書く，この贅沢。

大物プログラムを作ろうとすると，どうも腰が重くていけない。かなり気力が充実していないと，制作に取りかかれないのだ。なぜか。

管理が大変

これに尽きる。どこでどんな変数や関数を宣言して，どこで参照し，またその値を更新しているか，などという情報は，プログラムが大きくなればなるほど膨れ上がる。ましてやそれらすべてを矛盾なく整理するのは非常に骨の折れる仕事だ。

簡単に覚えきれる量ではないというのも問題だ。大きなプログラムの制作には時間がかかるので，いろいろと作っているうちに初めの頃に作ったものをすっかり忘れてしまう。それもよくない。

さらに不幸なことに，僕はソースリストに関してはきれい好きである。字下げには(自己流ながら)気を使うし，コメントの書き方にすら美しさを求める。未使用の変数の存在が許せないので，gccの-Wallオプションも欠かせない。

そういうわけなので，僕にとっては，覚えておかなければならない情報の多い大作プログラムを作るのは気の重い仕事なのだ。これから大きなプログラムを作ろうと考えたときに，自分が管理しなくてはならないものの量を考えただけで，どんよりとした気分になる。うじうじ考えているあいだに，えいっと作り始めればいいのだということはわかっている。いったん作業に取りかかればそんなにのろまではない(つもり)なのだが……と，なんだか愚痴みたいな文章だが，これは伏線。本題はこれからだ。

僕は，ずぼらな人間が大作プログラムを作り上げるための秘訣を最近ようやく身につけつつある。それこそが，

オブジェクト指向

である。

うーん怪しげ。こんな書き方を平気でしていると，近頃都に流行る怪しい宗教の伝道師みたいだ。「オブジェクト指向を使うだけで誰もが救われます」なんてね。

実のところ，オブジェクト指向は決して魔法の杖ではない，というより，「オブジェクト指向言語を使いさえすればすべての問題が解決する」などと考えてはならない。逆に，オブジェクト指向の志向するところを理解すれば，たとえオブジェクト指向言語を使わずとも，管理しやすいプログラムは書ける。要はちょっとした心掛けだ。ツールじゃなくて心なんだな。

つまみ食い，オブジェクト指向

大風呂敷を広げておいて畳むような真似になるが，本記事でオブジェクト指向の深淵に迫るつもりはまったくない。今回もC言語を使うつもりだからだ。

本格的オブジェクト指向は，やはりオブジェクト指向言語を導入したほうがやりやすいというものだ。クラスや継承のメカニズムは，C言語でも実現不可能ではないが，教材として見通しのよいものが書けそうな気がしない。

オブジェクト指向を身につけるための言語として，個人的にはC++に注目している(現在X68000上に実現されているC++に近い処理系にはGNUのg++がある)。C++は，特にオブジェクト指向を信奉する教条主義者にはけっこう評判が悪い。正しくオブジェクト指向していないというのが主な理由だ。まあ，それは当たっている。Smalltalkのような徹底したオブジェクト指向言語から見るとC++はいかにもいい加減で貧相なものに違いない。でも僕には「だから？」と平気でいい放つ用意がある。

この構図は，構造化プログラミングが叫ばれた頃のPascalとCの関係を思い起こさせる。「Cは高級言語でなく構造化アセンブラである」という名言もある。これを悪口ととるか最大級の賛辞ととるかはその人しだい。

C++は，オブジェクト指向言語というよりは，C言語にいくつかの便利な拡張を施し，オブジェクト指向のおいしい部分を取り入れたものであるという認識のほうがより実態を正確に把握している。それもまたよいではないか。求めて得た道具は使われる。従来のC言語の文法を破壊することなくオブジェクト指向っぽいプログラミングを可能にしているというところは，評価されていい。

＊　＊　＊

オブジェクト指向の正式な定義は知らない。僕が初めてオブジェクト指向らしいものを知ったのは数年前のことだが，そのときの定義はいまでもしっかりと心に残っている。それは，「オブジェクトが自分の動作を知っている」ということである。囲みの「カプセル化」の項を参照していただきたい。

オブジェクトは内部データとメソッドを自らの内部に隠し持ち，メッセージによってのみ行動する。このことは，変数や関数を局所的・分散的に管理することを可能にする。データとアルゴリズムをひっくるめたレベルのモジュール化である。

X Toolkitにおけるオブジェクト指向

いきなり話題を変える。

それにつけてもSX-WINDOWのプログラミングは面倒である。僕は本誌1991年1

月号で配布された開発キットが発表されてからSXアプリケーションを作りはじめたクチで，使うプログラミング言語ももっぱらCである。

「ウィンドウシステムのプログラミングは基本的に面倒なものだ。ある程度の作法を守って初めて見えてくる世界もある」という意見もあろう。それは正論。でもSX-WINDOWのプログラミングは面倒だとあえていいたい。どうしてそう思うかというと，もっとプログラミングの楽なウィンドウシステムを知っているからである。Macintoshではない(あれもSX-WINDOWと同じくらい面倒らしい)。X-Windowのプログラミングである。

X-Windowは，主にUNIXワークステーションで動作するウィンドウシステムである。もともとのOSが開発者を指向したシステムであるだけに，開発環境としての居心地のよさは他の追随を許さない。

X-Windowといったが，プログラミングを楽にしているのは「Xtイントリンシック」のこと。X-Windowそのもののライブラリである「Xlib」は低水準なもので，X-Windowのすべてをコントロールできる代わりにコーディングが厄介になってしまっている。XtイントリンシックはXlibを使って構築した上位ライブラリである。ウィンドウ上のプッシュボタンやプルダウンメニューなどを「ウィジェット」として部品化し，オブジェクト指向の概念を上手に導入している。これのプログラミング経験を持っている僕としては，SX-WINDOWのプログラミングはなんだかうっとうしく思えてしまうのだ。

XtはX Toolkitの略。これをパクって“SX Toolkit”の構想をぶちあげようというわけだ。

ウィンドウシステムはオブジェクト指向が向いているとされている。

画面に浮かぶウィンドウやその上のプッシュボタンはオブジェクト。ウィンドウやプッシュボタンをマウスでクリックしたりドラッグしたりする行為はオブジェクトに対するメッセージ。それに対するウィンドウやプッシュボタンの反応はオブジェクトの持つメソッド。

オブジェクトがどういうメッセージを受け取りどう身を処すべきかを知っているというのがオブジェクト指向なのであれば，ウィンドウやプッシュボタンがオブジェクトとして振る舞うのは自然なアプローチではあるまいか。

＊　＊　＊

Xtイントリンシックは現実にウィンドウシステムでオブジェクト指向的なアプローチに成功しているのだが，別にそれはOSがマルチタスクのUNIXだからだとかクライアント・サーバ・モデルの賜だとか，そんなことはない。ほかのウィンドウシステムでも十分できそうな感じがする。

まずは用語から。オブジェクト指向の用語はXtイントリンシックでは次のような用語で対応づけられている(あまり正確ではないが)。

オブジェクト＝ウィジェット
メッセージ　＝イベント
メソッド　　＝コールバック

この3つの単語をつなげると「ウィジェットはイベントを受け取りコールバックを発生する」となる。コールバックとは平たくいえば関数のことで，プッシュボタンが押される，などのような特定のイベントに対して特定の関数を実行させたい場合に使う。

通常，Xtは「ウィジェットセット」という，プッシュボタンやプルダウンメニューなどを定義するライブラリとともに使われる。そのひとつがOSF/Motif。囲みにごく簡単な例を挙げておいた。ウィジェットを作り，コールバックを登録しておくだけでよいというコーディングの単純さを見てほしい。たったこれだけのことをするためにもSX-WINDOWでは100行のコードを書き連ねる必要があるということも覚えておいてほしい。

注目したいのは，SX-WINDOWのイベントループに相当するXtMainLoop()。この関数を呼び出すだけで，あとはよきにはからってくれる。といっても，中でやっていることはSX-WINDOWのアプリケーシ

カプセル化

オブジェクト指向の特徴のうち，今回解説したいもの，それは「カプセル化」である。ありがちな例で恐縮だが，次のCプログラムを見ていただきたい。

```
double  x, y;
x = 2.0;
y = sqrt( x );
```

これが演算子指向(という用語があるのかどうか知らないが)プログラミングである。sqrt()という関数は，実数を引数に取り，その平方根を戻り値とする。この，実数を相手にするという情報や，平方根を求めるアルゴリズムは，関数sqrt()が知っている。xは単なる変数でしかない。

これをC++で書いてみる。まずは実数を表すクラスREALの定義。

```
class REAL  {
    double  value;
public:
    void    setvalue(double val)  {
              value = val; }
    double  squareroot()  {
              return sqrt(value); } ;
};
```

こうして定義したオブジェクトで平方根を求めてみる。

```
    REAL    x;
    double  y;
    x.setvalue(2.0) ;
    y = x.squareroot();
```

これは，オブジェクト指向風にいうと，「クラスREALに属する実数オブジェクトxに，自分の平方根を求めるというメッセージsquarerootを送ってその結果を受け取る」ということになる。うるさいことをいえばC++にはメッセージの概念はないし，ここでメッセージといっているのも単なるメンバ関数の呼び出しにすぎないのだが，少なくとも同じ概念のことが達成されている。

この例は単純なのでメリットがないように見える(オーバーヘッドが増える分不利ともいえる)が，実は効率を犠牲にしてもおつりがくるほどのうまみがちゃんとある。

squarerootというメッセージに対してどう対応すればいいかということは，実数オブジェクトしか知らないのである。実数クラスREALのメンバ関数として宣言したsquareroot()は，クラスREALに属する実数オブジェクトにしか適用できない。

たとえば文字列クラスSTRINGというものを，

```
class STRING  {
    char     *strptr;
public:
    void     setvalue (char *ptr)  {
               strptr = ptr; }
    int      length()  {
               return strlen (strptr); } ;
};
```

というふうに作ったとして，

```
STRING  s;
s.setvalue ("foo");
y = s.squareroot();
```

とやろうとしてもできっこない。文字列クラスにはメソッドsquarerootを定義していないからだ。

その逆もしかり。実数クラスの変数に対して文字列の長さを求めようとしてもムダである。当たり前だと思うだろうが，そうでもない。C言語にはこういうムチャなアクセスを防止する方法がないのである。

＊　＊　＊

つまりこれがカプセル化である。上の例では実数オブジェクトにはその値と平方根を求めるメソッドがひとまとめに入っている。文字列オブジェクトにはその先頭を指すポインタと長さを求めるメソッドが入っている。

カプセル化には，外からいじられたくない情報を隠蔽するという役割もある。上の例では，実数クラスREALの内部データvalueを直接読み書きできるのはメソッドであるsetvalue()とsquareroot()だけ。

カプセル化は，データを直接見せることなく，どこからでも利用できるようにするための，ある意味では画期的な手法である。

ョンとたいして変わらない。地道にループを組んで，マウスなどのイベントを取得し，そのイベントが対応するウィジェットを特定し，登録してあるコールバック関数を呼び出す。

SX Toolkitの構想

今回，SX Toolkitの試作システムを制作してみた。特集の趣旨も考慮して，あまりつっ走らずにSX-WINDOWのアプリケーションをオブジェクト指向的なアプローチで作ることを念頭に置いたものである。本当に味見というレベルなので，本格的に使うのはほとんど無理。が，真面目にライブラリ開発をやり通せば，SXlibの上の高水準ライブラリSXtが作れそうな感触を得た。

＊　＊　＊

基本的な戦略は，いわゆるスケルトンプログラムをX Toolkitライブラリを利用するのと同じ要領で使えるように分解してコーディングする。ではスケルトンプログラムを用意しておけばよいではないかという考え方もあるのだが，僕はそうは思わない。スケルトンと決定的に違うのは，ライブラリの中身についてアプリケーションプログラマは知る必要がないというところである。ソースリストもスッキリとし，ソースリストのイベントループ部分にありがちな妙に深いインデントもなくなる。

問題はコールバックで，多少トリッキーなコーディングが必要であった。SX-WINDOWでウィジェットにあたるものは「コントロール」として実現されているが，今回はその名もずばりWidgetという型を定義した。そのメンバには，

コントロールへのハンドル
コントロールへのID
コールバック関数のアドレス
コールバック関数に渡すパラメータ

などが入っている。

まずは仕掛け。ウィジェットを生成するときには，コントロールを内部的に作っているのだが，このときコントロールマンが用意しているユーザーデータ(コントロール構造体の中に定義してある)に，ウィジェットのアドレスを登録しておく。

コールバックを実現しているのがsxt.cのリストの326〜347行である。マウス左ボタンダウンイベントを取得してコントロールマンの関数CMFind()を呼べば，どのコントロールが選択されたかわかるので(ここまでは通常のスケルトンと変わらない)，そのコントロールのユーザーデータを参照して，ウィジェットのアドレスを得る。万一，そのコントロールがウィジェットとして作成されていなかった場合の危険性を考えて，マジックナンバーを調べるようにしてある(が，失敗すればバスエラーかなにかが出るだろうからたぶん役には立たない)。コントロールマンの関数CMCheck()を呼び出し，本当にボタンが押されたかを調べる(CMFind()で調べられるのは「ボタンを押した瞬間」であり，「ボタンを離すまで押されていたかどうか」はCMCheck()でフォローしてやる必要がある。こんな面倒なところはライブラリで吸収するのが一番)。そして，コールバック関数が登録してあれば，それを呼ぶ。

＊　＊　＊

SX Toolkitのライブラリ関数の構成や使い方は，X Toolkitのそれにわざと似せてある。コードは自前で書き起こしているので多少変な書き方もしていることだろう。

ウィジェットの生成パラメータの設定が少々回りくどい。ジオメトリ(座標や大きさ)を設定するのに，引数登録専用の変数を使わせている。これにはわけがある。ジオメトリだけを設定すればよいというのであれば，たとえば，

SXtCreateStdButton(win,x,y,w,h)

のように生成パラメータを引数リストに連ねるような形式の関数にすることも可能だろう。ただ，生成パラメータは今後増える予定であるし，ものによっては自動的にデフォルトの値を入れておけば十分ということもある(セレクトボタンの大きさなんてデフォルトの大きさで十分)。そうした可能性も想定すると，デフォルト以外の値だけを選択して指定できる，

SXtCreateStdButton(win,args,n)

の形式のほうが柔軟で，将来的には有利と考えた。Xtイントリンシックでは，リソース(SX-WINDOWのリソースとはかなり違う)との絡みでこの方式を採用しているようだが，SX-WINDOWとは直接関係を持たないので，リソースでウィジェット生成パラメータを設定することについては考えない。

＊　＊　＊

あくまで実験的なシステムである。不備はいっぱいだし，いくつかのひどい制限もある。

・起動時のコマンドラインオプションがユーザーに見えない。
・標準ボタンしか用意していない。
・ユーザープログラムにはウィジェットに対するコールバック以外のかたちでしかイベントがこない。ヌルイベントやアップデートイベントなどに対するコールバックをサポートしないと，たいていのアプリケー

Xt/Motifのプログラミング

Xtイントリンシックの典型的なプログラミング例を次に示す。概念を示すにとどめるので，必要な関数の一部を省略している。そのままでは動作しない。

もちろんX68000上では動作しない。

(1)toplevelはひとつのアプリケーションにひとつ作るおおもとのウィンドウ
(2)bulletinboardはプッシュボタンなどを張りつけるための板
(3)pushbuttonはプッシュボタン
(4)プッシュボタンにコールバックを登録する。ボタンが押されたらfooCBを呼ぶように設定している
(5)メインループ。正体はイベント待ちの無限ループで，イベントを処理し，適切なコールバックを発生させる。

これは，プッシュボタンのひとつついたウィンドウを作成し，そのボタンを押すと「押された」と標準出力にプリントするプログラムである。

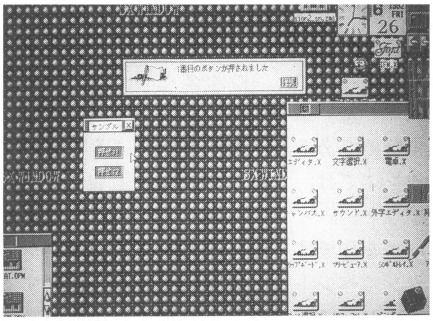
```
void    fooCB( w, client_data, call_data )
Widget  w;
caddr_t *client_data, *call_data;
{
    printf( "押された¥n" );
}


void    main( argc, argv )
{
    Widget    toplevel, bulletinboard, pushbutton;


    toplevel = XtInitialize( argv[0], "Sample", NULL, 0, &argc, argv );          (1)
    bulletinboard = XmCreateBulletinBoard( toplevel, "Bulletinboard", NULL, 0 );  (2)
    pushbutton = XmCreatePushButton( bulletinboard, "Pushbutton", NULL, 0 );      (3)
    XtAddCallback( Pushbutton, XmNactivateCallback, fooCB );                      (4)
    XtMainLoop();                                                                 (5)
}
```

ションは書けないだろう。
・キーダウンイベントがいい加減。自分がアクティブウィンドウであるときにリターンキーが押されたら，終了してしまう。
・アイコン化をサポートしていない。
・ウィジェット変数の管理がいい加減。配列で確保しているので登録はできても削除はできないし，必要にして十分な量だけを利用することができない(これは教材と割り切ってのこと)。
・複数起動できない。したら終了時にシステムエラーを食らう(これはウィジェット変数の管理がいい加減なせい)。
・以上の理由から本格的には使えない

ないないづくしだが，どれも解決可能な問題ではある。もっとポジティブに見ることにしよう。SX Toolkitはスケルトンでなくライブラリだからブラックボックスで扱える。ライブラリが整備されれば，リストに掲げたsample.cのように，ほんの数十行でいっぱしのウィンドウが開けるようになるはずだ。

けっこう面倒見はよくしたつもり。起動時はコマンドラインのパラメータを見て-Wスイッチがあればそのとおりにウィンドウを開き，なければ適当なところに開く。イベントループ中も，ウィンドウをドラッグする処理も気にしなくていいし，クローズボタンが押されれば勝手に終了処理に飛ぶし，アップデートイベントがくれば少なくともウィジェットは描き直す。コールバックだけが外に出てくる。終了時はウィジェットとウィンドウを勝手に廃棄してくれる。処理のキモの部分だけをプログラマは気にすればいい。ただし，それには，グローバル変数を同名複数のタスクで共有してしまうという現在のSX-WINDOWのタスク管理はきつい障害となりそうだ。

SX Toolkitが完成した暁には，片っ端からウィジェットを作って片っ端から動作(コールバック)を教え，あとは寝て待つだけというぐうたらなプログラミングスタイルが可能になる。


サンプルプログラムの実行例

ウィジェットを作り，それのコールバック関数を登録すると，あとは放っておいてもシステムが面倒見てくれる。これがオブジェクト指向を採用するひとつの大きな魅力だ。作ったウィジェットが自分のすべきことを知っていて，自分に関係のありそうなイベントだけを取ってきて適切な反応を返す。わざわざイベントループを組んでイベントがどのコントロールに対応するもの

SX Toolkit 試作版・関数リファレンス

void SXtSetArg(Arg *arg, int type, int value)

ウィジェットやウィンドウを作成するための引数をセットする。
typeには次のいずれかを与える。その場合のvalueの型も併記する。

SXtArgLabel	ラベル	char *
SXtArgX	x座標	int
SXtArgY	y座標	int
SXtArgW	幅	int
SXtArgH	高さ	int
SXtArgWdefID	ウィンドウ定義関数のID	int

window *SXtInitialize(Arg args, int n);

初期処理。アプリケーション起動直後に呼び出す。
ウィンドウを開いてそのポインタを返す。

Widget SXtCreateStdButton(window *win, Arg args [], int n)

親ウィンドウ win の下に標準ボタンを作る。

Widget SXtSetSelectCB(Widget wd, void (*procPtr)(), int data)

ウィジェットwdが選択されたときに発生させるコールバック関数を登録する。
コールバック関数はユーザーが用意する関数で次のように書く(関数名は任意)。

```
int    procPtr( Widget wd, int data )
 {
       ・
       ⋮
       return;
 }
```

void SXtRealize(window *win)

ウィンドウwinをリアライズする(可視にする)。
ウィジェットも同時に描画する。

void SXtMainLoop(window *win)

イベントループ処理。
ウィジェット，アップデート，タスクマンのイベントを取り扱う。
現時点では，
・クローズボタンが押されたとき
・タスク終了イベントを受け取ったとき
に終了する(下のSXtExitを()呼ぶ)。

volatile void SXtExit(window *win, int endCode)

アプリケーションを終了する。
このとき作成したウィジェットとウィンドウwinを廃棄する。
通常は SXtMainLoop() から終了時に呼ばれる。
なお，ユーザープログラムがexit()関数を呼んで終了してはならない。
(作成したウィジェットが廃棄されないで残るため)

リスト1

```
/*
 *    sample.c
 *    - SXt のサンプルプログラム
 *      1992/06/21   丹 明彦
 */

#include    "sxt.h"

void    pushedCB( wd, data )    /* コールバック関数 */
Widget wd;
int     data;
{
    char    tmp[32];
    void    sprintf();

    sprintf( tmp, "%d番目のボタンが押されました", data );
    DMError( 0x0001, tmp );

    return;
}

void    main()
{
    Arg      args[5];
    Widget   pushme1, pushme2;
    window   *toplevel;

    /* ウィンドウを開く */
    SXtSetArg( &args[0], SXtArgW, 90 );
    SXtSetArg( &args[1], SXtArgH, 90 );
    SXtSetArg( &args[2], SXtArgLabel, "サンプル" );
    toplevel = SXtInitialize( args, 3 );
    /* ウィジェット(ボタンその1)を作成する */
    SXtSetArg( &args[0], SXtArgX, 20 );
    SXtSetArg( &args[1], SXtArgY, 20 );
    SXtSetArg( &args[2], SXtArgW, 50 );
    SXtSetArg( &args[3], SXtArgH, 20 );
    SXtSetArg( &args[4], SXtArgLabel, "押せ:1" );
    pushme1 = SXtCreateStdButton( toplevel, args, 5 );
    SXtSetSelectCB( pushme, pushedCB, 1 );
    /* ウィジェット(ボタンその2)を作成する */
    SXtSetArg( &args[1], SXtArgY, 50 );
    SXtSetArg( &args[4], SXtArgLabel, "押せ:2" );
    pushme2 = SXtCreateStdButton( toplevel, args, 5 );
    SXtSetSelectCB( pushme, pushedCB, 2 );
    /* ウィンドウをリアライズし，ウィジェットを表示する */
    SXtRealize( toplevel );
    /* イベントループ */
    SXtMainLoop( toplevel );
}
```

かを調べるなどという行為はもういらない。

プログラムのメンテナンス性も大幅に向上する。ウィジェットを追加したい場合や減らしたい場合に，いじる場所が少なくてすむのである。スケルトンによるプログラミングでは，コントロールを追加しようと思うと，ソースリストのうち，(1)コントロールを作成する場所，(2)マウスのボタンダウンイベントがどのコントロールに対するものかをチェックして処理する場所，(3)コントロールを廃棄する場所，の3カ所をいじらなくてはならない。その3カ所は離れたところにあるので，書き換えがめんどうくさい。これに対して，ウィジェットを使っているものは，(1)ウィジェットを作成する場所，と(2)コールバックを登録する場所，が1カ所にまとめられるし，(3)ウィジェットを廃棄する場所，はない(ライブラリでタスク終了時に勝手にやってくれる)。ひとつのウィジェットに対するコードが1カ所に集中しているので，書き換えが楽なのだ。

というわけで，冒頭に出てきていた大作プログラムの話とつながる。C言語でも，変数をオブジェクトの内部データ，関数をメソッドのように振る舞わせるというコーディング作法は，局所的かつ分散的なデータ管理を実現する。局所的だから内部構造を変更しても影響は最小限だ。グローバル変数を直接いじるような野蛮なやり方ではとうてい実現できない世界がそこにはある。そう，キャッチフレーズは「作り散らかしても大丈夫」。

参考文献

X-Window OSF/Motif プログラミング
兜木昭男/木下凌一/栄谷政己/林秀幸/安川悦子著，日刊工業新聞社

図1 SXlibとSXtoolkitのメンテナンス性

SXlib　　　　　　　　修正箇所

```
main() {
  ....
  button1 = CMOpen( ... );
  button2 = CMOpen( ... );                 コントロール作成
  ....
  while ( 1 ) {
    ....
    switch ( eventRec.what ) {
      ....
      case E_MSLDOWN:
        ....
        hndl = CMFind( ... );
        if ( hndl == button1 ) {
          ....
          CMCheck ( ... );
          ....
        }
        if ( hndl == button2 ) {           イベント処理
          ....
          CMCheck ( ... );
          ....
        }
        ....
    }
  }
}

endTask() {
  ....
  CMDispose( button1 );
  CMDispose( button2 );                    コントロール廃棄
  ....
}
```

SXtoolkit　　　　　　　　修正箇所

```
main()
{
  ....
  button1 = SXtCreatePushButton( ... );
  SXtSetSelectCB( button1, ... );
  button2 = SXtCreatePushButton( ... );    ウィジェット作成
  SXtSetSelectCB( button2, ... );          コールバック登録
  ....
  SXtMainLoop( ... );                      (この中はいじらない)
}
```

アミ掛け部を修正する

リスト2

```
/*
 *   sxt.c
 *   - SX-WINDOW toolkit: Xt (X toolkit) のまねごと
 *       1992/06/21   丹 明彦
 */

#include   <string.h>
#include   "sxt.h"

/*
 *  アプリケーションで使うウィジェットの実体
 *  - 手抜きだが配列で宣言している
 */

#define    MAXWIDGET   100

WidgetS    widget[ MAXWIDGET ];
int        nWidget = 0;


/*  ウィンドウとウィジェットの定義パラメータ */

int  sxtarg_x, sxtarg_y, sxtarg_w, sxtarg_h, sxtarg_wdefID;
LASCII sxtarg_label;
rect    sxtarg_geom;

/*
 *  void setArgs( Arg args[], int n )
 *  - 指定された引数をセットする
 *     外部からのアクセスは禁止
 */

void  setArgs( args, n )
Arg    args[];
int    n;
{
    int   i;

    for ( i = 0; i < n; i++ ) {
      switch ( args[i].type ) {
      case SXtArgLabel:
          strcpy( sxtarg_label.Lstr, (char *)(args[i].value) );
          sxtarg_label.length = strlen( sxtarg_label.Lstr );
          break;
      case SXtArgX:
          sxtarg_x = (int)(args[i].value);
          break;
      case SXtArgY:
          sxtarg_y = (int)(args[i].value);
          break;
      case SXtArgW:
          sxtarg_w = (int)(args[i].value);
          break;
      case SXtArgH:
```

```
            sxtarg_h = (int)(args[i].value);
            break;
        case SXtArgWdefID:
            sxtarg_wdefID = (int)(args[i].value);
            break;
        default:
            break;
        }
    }
    return;
}

/*
 *  void setGeometry()
 *  - ジオメトリをレクタングル sxtarg_geom に格納する
 *    外部からのアクセスは禁止
 */

void   setGeometry()
{
    sxtarg_geom.left   = sxtarg_x;
    sxtarg_geom.top    = sxtarg_y;
    sxtarg_geom.right  = sxtarg_x + sxtarg_w;
    sxtarg_geom.bottom = sxtarg_y + sxtarg_h;

    return;
}

/*
 *  void SXtSetArg( Arg *arg, int type, int value )
 *  - ウィジェットやウィンドウを作成するための引数をセットする
 */

void   SXtSetArg( arg, type, value )
Arg    *arg;
int    type, value;
{
    arg->type = type;
    arg->value = value;

    return;
}

/*
 *  window *SXtInitialize( Arg args, int n );
 *  - 初期処理
 *    アプリケーション起動直後に呼び出す
 *    ウィンドウを開いてそのポインタを返す
 */

window *SXtInitialize( args, n )
Arg     args[];
int     n;
{
    int     paramStatus;
    int     *paramBuf;
    window *win;
    task    taskBuf;
    int     argc;
    char    **argv;

    /* タスク管理テーブルを得る */
    TSGetTdb( &taskBuf, -1 );
    /*
     *  パラメータを解析する
     *  -  "-W"などのシステムパラメータをここで取り出す
     */
    paramStatus = TSTakeParam( &taskBuf.command, &sxtarg_geom,
                               0, 1, 0, &paramBuf );
    argc = (int)(paramBuf[0]);
    argv = (char **)(&(paramBuf[1]));
    /*
     *  コマンドラインを消去する
     *  - 次回起動のときにゴミを残さないため
     */
    taskBuf.command.Lstr[0] = '¥0';
    taskBuf.command.length = 0;
    TSSetTdb( &taskBuf, -1 );
    /* デフォルト値を設定する */
    sxtarg_wdefID = 49;
    if ( (paramStatus & 1) == 0 ) {
        /*
         *  位置の指定がない場合
         *  - ウィンドウサイズはデフォルト
         */
        *((long *)(&sxtarg_geom.left)) = TSGetWindowPos();
        sxtarg_x = (sxtarg_geom.left);
        sxtarg_y = (sxtarg_geom.top);
        sxtarg_w = 150;
        sxtarg_h = 100;
    } else {
        /* 位置の指定がある場合 */
        sxtarg_x = (sxtarg_geom.left);
        sxtarg_y = (sxtarg_geom.top);
        sxtarg_w = (sxtarg_geom.right)  - (sxtarg_geom.left);
        sxtarg_h = (sxtarg_geom.bottom) - (sxtarg_geom.top);
    }
    strcpy( sxtarg_label.Lstr, "window" );
    sxtarg_label.length = strlen( sxtarg_label.Lstr );
    /*
     *  引数の解析
     *  - 上で設定したジオメトリが上書きされることもある
     *    つまりハードコーディング優先ということ
     */
    setArgs( args, n );
    setGeometry();
    /* ウィンドウを開く(不可視,クローズボタンあり) */
    win = WMOpen( NULL, &sxtarg_geom, &sxtarg_label, FALSE,
        sxtarg_wdefID<<4, (window *)-1, TRUE, TSGetID() );
    /* アボート終了処理を登録 */
    TSSetAbort( (void *)&SXtExit, (long)win );

    return ( win );
}

/*
 * Widget SXtCreateStdButton( window *win, Arg args[], int n )
 *  - 親ウィンドウ win の下に標準ボタンを作る
 */

Widget   SXtCreateStdButton( win, args, n )
window   *win;
Arg      args[];
int      n;
{
    Widget   wd;

    /* デフォルトの値をセットする */
    sxtarg_x = 0;
    sxtarg_y = 0;
    sxtarg_w = 40;
    sxtarg_h = 20;
    strcpy( sxtarg_label.Lstr, "StdBtn" );
    sxtarg_label.length = strlen( sxtarg_label.Lstr );
    /* 指定された引数をセットする */
    setArgs( args, n );
    /* 大きさをレクタングル geom に格納する */
    setGeometry();
    /* ウィジェットをひとつ確保する */
    wd = &widget[ nWidget++ ];
    /* マジックナンバー */
    wd->magicNo = SXTMAGICNO;
    /* ボタンを作成する */
    wd->ctrlHdl = CMOpen( win, &sxtarg_geom, &sxtarg_label, TRUE,
            1, 0, 1, CI_STDBTN<<4, (long)wd );
    wd->ctrlID = CI_STDBTN;
    /*
     *  コントロールのユーザーワークにウィジェットのアドレスを入れる
     *  - コールバック先の探索を楽にする
     */
    CMUserSet( wd->ctrlHdl, (long)wd );
    /* コールバックはデフォルトで未定義 */
    wd->selectCB = NULL;
    /* 作成したウィジェットを返す */
    return ( wd );
}

/*
 * Widget SXtSetSelectCB( Widget wd, void (*procPtr)(), int data )
 *  - ウィジェット wd が選択されたときに発生させるコールバック関数を登録する
 *    コールバック関数はユーザーが用意する関数で次のように書く(関数名は任意)
 *
 *    int   procPtr( Widget wd, int data )
 *    {
 *       .....
 *       return;
 *    }
 */

void     SXtSetSelectCB( wd, procPtr, data )
Widget   wd;
void     (*procPtr)();
int      data;
{
    wd->selectCB = procPtr;
    wd->selectCBdata = data;

    return;
}

/*
 *  void SXtRealize( window *win )
 *  - ウィンドウ win をリアライズする(可視にする)
 *    ウィジェットも同時に描画する
 */

void     SXtRealize( win )
window   *win;
{
    WMShow( win );
    GMSetGraph( &(win->wGraph) );
    CMDraw( win );

    return;
}

/*
 *  void SXtMainLoop( window *win )
 *  - イベントループ処理
 *    ウィジェット,アップデート,タスクマンのイベントを取り扱う
 */

```

```
void    SXtMainLoop( win )
window   *win;
{
    int   active = TRUE;/* アクティブウィンドウかどうか */
    tsevent   eventRec; /* イベントレコード構造体 */
    control   **selHdl; /* セレクトされたコントロール */
    Widget   wd;        /* セレクトされたウィジェット */
    point_t   pt;       /* マウスイベントの発生した座標 */

    while( 1 ) {
        TSEventAvail( EM_EVERY, &eventRec );
        switch ( eventRec.what ) {
        case E_IDLE:          /* アイドルイベント */
            break;
        case E_SYSTEM1:       /* システムイベント */
        case E_SYSTEM2:
            switch ( eventRec.what2 ) {
            case SAVE:        /* パラメータのセーブ */
                break;
            case CLOSEALL:    /* 全クローズ */
            case ENDTSK:      /* タスク終了 */
                SXtExit( win, 0 );
                break;
            case WINDOWSELECT:/* ウィンドウセレクト */
                WMSelect( win );
                active = TRUE;
                break;
            }
            break;
        case E_UPDATE:        /* アップデートイベント */
            WMUpdate( win );
            GMSetGraph( &(win->wGraph) );
            CMDraw( win );
            WMUpdtOver( win );
            continue;
        case E_ACTIVATE:      /* アクティベートイベント */
            if ( (window *)(eventRec.whom) == win ) {
                WMSelect( win );
                active = TRUE;
            } else {
                active = FALSE;
            }
            break;
        case E_MSRDOWN:       /* マウス右ボタンイベント */
            /* イベントは自分のウィンドウに対するものか */
            if ( (window *)eventRec.whom != win ) break;
            /* アクティブウィンドウにする */
            if ( active == FALSE ) {
                WMSelect( win );
                active = TRUE;
            }
            break;
        case E_MSLDOWN:       /* マウス左ボタンイベント */
            /* イベントは自分のウィンドウに対するものか */
            if ( (window *)eventRec.whom != win ) break;
            /* アクティブウィンドウにする */
            if ( active == FALSE ) {
                WMSelect( win );
                active = TRUE;
            }
            /* この時点までボタンが押されているか */
            if ( EMLStill() == 0 ) break;
            /*
             * ウィンドウアイテムの処理
             * - ウィンドウのドラッグなど
             */
            switch ( SXCallWindM( win, &eventRec ) ) {
            case W_INCLOSE:   /* クローズボタンが押されたら終了*/
                SXtExit( win, 0 );
            default:   /* コントロール(ウィジェット) */
                GMSetGraph( &(win->wGraph) );
                pt.x_y = eventRec.whom2;
                pt.x_y = GMGlobalToLocal( pt );
                CMFind( pt, win, &selHdl );
                /*
                 * コントロールに対応するウィジェットを得る
                 * - マジックナンバーを見て対応するウィジェット
                 *   が存在することを確認する
                 */
                wd = (Widget)CMUserGet( selHdl );
                if ( wd->magicNo != SXTMAGICNO ) break;
                switch ( wd->ctrlID ) {
                case CI_STDBTN:   /* 標準ボタン */
                    if ( CMCheck( selHdl, pt, NULL ) == 0 ) break;
                    /* コールバック関数(登録してあれば)を呼ぶ */
                    if ( wd->selectCB != NULL )
                        wd->selectCB( wd, wd->selectCBdata );
                    break;
                }
            }
            break;
        case E_KEYDOWN:       /* キーダウンイベント */
            if ( active == FALSE ) break;
            /* とりあえずリターンキーで終了するようにしている */
            if ((short)(eventRec.whom) != 13 ) break;
            SXtExit( win, 0 );
            break;
        }
    }
}

/*
 *  volatile void SXtExit( window *win, int endCode )
 *  - アプリケーションを終了する
 *    作成したウィジェットを廃棄する必要があるので
 *    ユーザープログラムが勝手に exit() してはならない
 *    通常は SXtMainLoop() から終了時に呼ばれる
 */

volatile void   SXtExit( win, endCode )
window   *win;
int   endCode;
{
    int   i;
    void   exit();

    /* 作成したウィジェットを廃棄する */
    for ( i = 0; i < nWidget; i++ ) {
        CMDispose( widget[i].ctrlHdl );
    }
    /* ウィンドウを廃棄する */
    WMDispose( win );

    exit( endCode );
}
```

リスト3

```
/*
 *  sxt.h
 *  - SX-WINDOW toolkit
 *    Xt (X toolkit) のまねごと
 *    コントロールを"ウィジェット"として扱う
 *    ウィジェットへのイベントはコールバックとして処理する
 *    アプリケーションは局所的・分散的にコーディングできる
 *    1992/06/21   丹 明彦
 */

#define   __POINT_T
#include  <sxlib.h>

#define   NULL   0

#define   FALSE   0
#define   TRUE    ~(FALSE)

/*
 *  マジックナンバー
 *  - ウィジェットでないコントロールからのアクセスを防止する
 */

#define   SXTMAGICNO   'SXTK'

/* ウィジェット構造体 */

typedef   struct {
    long      magicNo;
    control   **ctrlHdl;
    int       ctrlID;
    /* コールバック関数のアドレスとコールバック関数に渡る値 */
    void      (*selectCB)();        /* ボタンが押されたとき */
    int       selectCBdata;
} WidgetS;

typedef WidgetS   *Widget;

/* ウィジェットを生成する引数を格納する構造体 */

typedef struct {
    int  type;
    void *value;
} Arg;

/* 引数のタイプ */

#define  SXtArgLabel  0  /* ラベル char *          */
#define  SXtArgX      1  /* x座標  int             */
#define  SXtArgY      2  /* y座標  int             */
#define  SXtArgW      3  /* 幅     int             */
#define  SXtArgH      4  /* 高さ   int             */
#define  SXtArgWdefID 5  /* 種類   int ウィンドウ定義関数のID */

/* SX-WINDOW toolkit の関数(ユーザーがアクセスしていいもの) */

void       SXtSetArg( Arg *, int, int );
window     *SXtInitialize( Arg *, int );
Widget     SXtCreateStdButton( window *, Arg *, int );
void       SXtSetSelectCB( Widget, void (*)(), int );
void       SXtRealize( window * );
void       SXtMainLoop( window * );
volatile void   SXtExit( window *, int );
```

比較的大きなプログラムの独断的制作法

ちょっと大きいモノを書こう

Yokouchi Takeshi 横内 威至

大きめのプログラムを作るときの手順や効率のよい方法はなにか？ ゲーム作りのノウハウを交えながら，プログラム作成の流れをSM.Xを例に取って解説してみましょう。

序章

「最近のソフトはスゲーよな」
「Gなんとかなんてスゲーよな」
「でももう飽きたからさっさと次のやつ作ってくれよなー」
「もうやることないから次はFINALなんとかってやつでも待つか」

こんなこといってる人たちはさっさとゲーセンでも行きなさい。

「Gなんとかなんてスゲーな」
「だいたいデカいのあんなに動かすのにスプライト128個でよく足りるよな」

そうだ。いいぞ。だがもうちょいとだ。

「あれが出るってことはスト○もいずれは……。でもオレは人間は取り込みで，死体は血を流して横たわっていてリンチの快感が味わえないと許さん。いつかは作ってやるがどうすればいいかわからん」

こいつだ。まずなにか作ろうというには野望が必要だ。野望をお持ちの方は多数いることと思う。だがその先でつまずくことがほとんどであろうとも思う。OKだ。巨大なプログラムを作るための私なりのステップを，貴方の野望の参考のために聞いてもらおう。

STEP1-鍛錬

これは小さいプログラムぐらいなら組める方に有効な話だ。そのレベル以前の人はまず言語をマスターすることだ。「マスター」というと，いいすぎだが，まあ本なんかを見ずにプログラムがある程度書けるぐらいにはなっておこう。

そして，言語とともに風土を知らねば生きていけない。ハードウェアも当然知らねばならない。本誌の連載なんかは素晴らしいバイブルだ。しっかり勉強しておこう。

このくらいわかるという方なら，もう十分である。すでに皆さんのハードでどんなことができるかというのはだいたい見えているだろう。では，本題に入っていってみようか。

企画

実は，ここがもっとも難しいはずだ。やりたいことをしっかり把握し，まず全体像を考えよう。まあ，企画ぐらいは皆できるはずであるが，無謀な企画もあるはずだ。軽くチェックを入れてみよう。

その1　**ハードをよく見極めろ**

X68000は確かに素晴らしいハードを持っている。とはいっても，無理なことは無理なのである。ハードの制約を考慮しないといけないのだ。だからこそ厳しい。ハードを知っていればなにをなんのためにどう使うかを考えながら仕様を決定していける。

実際に作られたソフトウェアのなかには，一見，X68000のハードを越えているようなものがあるのは事実だ。でも，それにはちゃんと裏があるものだ。チョロっと明かしてしまおう。

まずアフターバーナーだ。地上物が少ないのはなぜだと思う？ 実はスプライトなのだ。背景の回転，地平線の移動もいちいち書き換えなどするわけにはいかない。パレット＋スクロールだ。よくわからないならリセットしてグラフィックを見てみよう。似た例ではあるがサンダーブレードも建物や戦車なんかはスプライト，地上の道なんかはテキストだ。

G2やFINAL FIGHTなんかも大きいからといってグラフィックではない。あらゆるスプライトを1回1回書き換えているのだ。まあ，だからといって簡単ではない。当然，今度はソフトの技術が必要なのだ。

どうだ？ ハードがわかっていればこのようなシブい使用法が浮かぶのである。そう，最初に仕様をそれなりに決めないとそれぞれの部分をどのように活用させるか見えてこないのだ。ゲームしか例に挙げてないがなんでもそうなのである。

その2　**よきモノは見習え**

企画内容自体についてはなにもいうまい。ただし常に客観的な視点からも見つめなければならない。人様に決して渡ることのないものならよいが，そうでないなら先人の道を常に考慮しなければならない。気の狂ったフォーマットや操作はポリシーなしにはすべきではない。

そしてもうひとつ，ユーザーよりもプログラマが苦汁を味わうべきなのだ。これをこうすれば楽になるのはわかるがプログラムが面倒臭い，などとほざくことのないように。

その3　**記録は残せ**

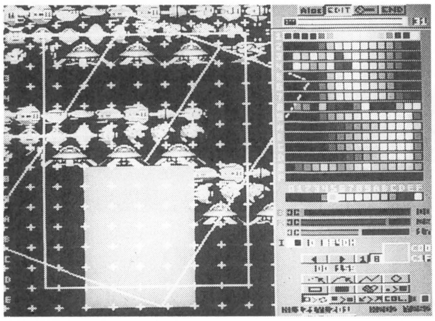
さて，概要が見えたらすべて紙に書き写すことを当然おすすめしよう。画面割だとかデータフォーマットだとか細かい仕様だとか思いつくものすべて書くのだ。当たり前のことではあるが。どんなにショボいものでも頭で考えるよりは役立つ。

＊　＊　＊

さて，ハードを考えながら企画していくのだが，具体例を示したほうがつかみやすいと思う。あまりよい例ではないのだが，6月号のディスクのおまけのSM.Xをさらしものにする。なぜ悪い例かというと，予定からガンガンはずれたし，アセンブラもまだハンパだし，アルゴリズムはカスだし，無駄だらけだし，ソースも汚いし，そのため異様に長いし，その他いろいろだからである。

＊　＊　＊

まず，最初に漠然とスプライトエディタがほしかった。で，世の中になにがあったかといえば，オマケのBASICのヤツ。そのほかにもまああったが，ダメ。なぜか。でかいモノがあれじゃやりにくい。かつてGなんとか（前に述べたヤツじゃない）のと



お馴染みSM.X

きに痛感したことだ。

そして単純な回転もあれば便利だ。となれば，作業すべき領域はグラフィックがよい。そうすればほかで描いたモノを利用することもできる。ゲームのためがメインだから256×256のほうがよい。編集のためにカット&ペーストで別のエリアもあったほうがよい。

もうちょっと深く考えよう。でかいのをドット単位でやるのは馬鹿だ。では，よくあるグラフィックエディタを真似しよう。だが，ルーペも同じく重要だ。それならカット&ペーストも含めてモードを3つに分けてしまおう。

エリアは256×256で編集用と作業用の2枚でジャストだ。じゃあ余ったテキストをメニューに。そういえば指定枠なんかはグラフィックだな。2枚のグラフィックの片方の余ったところでOKだ。色指定は本当はグラデなんかがほしいが16色だからだめだ。これは普通にいくか。ただし16色で色選びをわざわざカーソル移動でやるのは頭が悪い。左手が空くのは見えているからキーボードを使ってやろう。よし，だいたいいいや。とりあえず一服しよう。

*　　*　　*

こんな感じだ。かなりアバウトだがこんな程度のものはこれでいい。あくまでも1例にすぎないが，もっと真面目なもの，たとえばワープロだとかなんかはデータのフォーマットなんかは素晴らしく重要だろう。

プログラミング

そろそろプログラムについて考える。企画が揃ったところでどうやっていくかが，なかなか難しいものであろう。本来あなたなりのスタイルがあると思うが，ない人はやっぱり参考にしてほしい。

●アルゴリズム

これは私の思うことだが，プログラムは大きく4つに分けられると思う。まずひとつ目はベースとなるメインループ，2つ目は全体の下請け作業を受け持つサブルーチン，3つ目はデータ，4つ目は独立して動ける専門サブルーチンだ。

最初のヤツはサブルーチンコールが並ぶ基本ループのこと，2つ目はIOCSのようなI/O制御部や1文字プリントルーチンなんか，4つ目は2つ目とあまり差はないが，大きくなればMAGICのような3D計算部分なんかのことである。本当はあとひとつ，割り込み用のプログラム，MUSICドライバなんかも存在するのではあるが，まあいいだろう。

まず作るべきは4つ目，独立ルーチンとそれが使うサブルーチンである。これはもう皆さんなら可能であろうし，MAGICのように既存のものを流用してもいいだろう。もちろん，そのためのデータフォーマット，コール時のレジスタ，使用するレジスタなどはしっかり控えておくこと。OKだよな，みんなわかっているよな。

そして次はメインループだ。サブルーチンはかなり細かく，多くなるのでまずマクロな部分を簡単に考える。フローチャートというヤツを書くといいだろう。細かく書く必要はない。アバウトにやるのだ。流れさえつかめばよいのである。

くどいがサブルーチンなんかあとでいい。というのは，本体を作っていかないとどんなサブルーチンが必要で有効かは見えてこない。また，よほど綿密な計画でもしない限り絶対にあとからつけ足していかなければならないからだ。そしてバグも出やすく，いずれ高速化なんかもしないとならないし，いきなり細かいことで悩むのはヘビーすぎるからである。

●試し打ち

アルゴリズムの次はいよいよ実際に打ち込んでやるのだ。ちょっとした，目に見える効果のあるルーチンだけを，そして影響を及ぼす適当なルーチンをあわせてテストのようなプログラムから入っていくとよい。とにかく目で見て変化があるようになるところまでは作ったほうがいい。

サブルーチンは前述のとおり，さほど作ってないなら“RTS”なんかで置き換えといてやる。またはデータがちゃんと渡されているかを確かめるため，表示だとかしてやってもよい。

また同じようなサブルーチンを過去に作っており，ライブラリにストックしてあるなら引っぱってきなさい。これから作るサブルーチンなんかもいずれはライブラリにつけ足すことになるので，フォーマットやコール方法，そしてソースリストにはちゃんとわかるようにドキュメントを書いておいてやることが重要である。

さて，狙ったように動作してくれたであろうか。まだこれではそれほどバグはでないと思う。出たところで見える動作の中でのバグだ，デバッガなんかなくてもすぐ直せるはずだ。

●サブルーチン補充

まあ，前節でやっていてもよいが基本的なサブルーチンなんかをさっさとつけてやる。画面クリアだとか簡単な表示だとかをだ。するとすでに，あなたの望むような光景が画面上に拡がってきていることになるはずである。あとはこの調子で一気に進むだけなのである。

残っているのは細かいサブルーチンとデータであるが，すでに形のあるこのプログラムにつけ足すのは容易なこととなるであろう。技術が上がれば高速化もガンガンしてやりなさい。ここからは皆さんの頭と力の見せつけどころとなっているのである。

⋮

ここまで読んで異常だと思う方も多いと思う。おそらく一般とは逆の作業行程だからである。かつて，あの古籏一浩氏が語った方法ではまずサブルーチンを完成させ，集め，そしてメインルーチンであった。だが私はまったく逆なのである。なぜなのだろう，ちょっと考えてみよう。

まず私は，性格からして精密な計画を練り上げることができないのである。とりあえず思い浮かぶ光景を画面に築き上げて，初めて必要なものが浮かび上がってくるのだ。

また，私の方法の，よいと勝手ながら思うところは，独立して独り歩きしそうな部分をまわりの部分と照らし合わせて調整しながら作っていけるところにあると思っている。

そしてなにより，サブルーチンはひとつずつ，あとから足していくので，バグが出たときはたいていそこだけのバグで収まるのである。となれば，デバッガなんか使うことなく容易にバグ取りができるのだ。実際，いままでにデバッガを使ったことは一度たりともない。

ということで，私のやり方は大きな絵を描くようなものと思ってほしい。まずインスピレーションがわいたら全体をぼんやりと想像し，簡単なデッサンを作成する。そしてキャンバスに概形をさっさと描写し，とりあえず目で確かめることで全体を確かめ直すのだ。そしていよいよ細かいところをネットリと描いていく。ほかとのバラン

スを保ちつつ，浮かばせるところは浮かばせ，ミスったら修正しながら……。

そうである。細かい絵を先に，女A，馬A，山A，などを先に完成させて，ラストでそれらを組み合わせるよりもおそらく全体を見ながらやっていけると私は思うのだ。だから私はこれでよいと思っている。皆さんにもおすすめする，強要はしないが。

SM.Xはこうして作られた

例によってSM.Xの進行過程を簡単に示しておこう。ただしあまり参考にならないかもしれない。私もここまで悲惨な作り方をしたのは初めてだった。いい加減さがしんみりと伝わることだろう。

1)　初期画面設定ができて走らす。画面の絵ができた。カーソルはまだない。

2)　ウィンドウもどきのフォーマットを考え，なにもしてくれないウィンドウをとりあえず置いておきカーソル部分（モード1用）を作った。

3)　色のあたりを考える。キーボードはまだ使えない。数値表示もまだしてくれない。ついでにPSETを作って点画が描けるようになった。

4)　寂しいので数値表示や簡単なスクロールをつけてやる。全体にからまない罫線なんかもつけてやる。

5)　ラインを作り，指定線を使うモノの代表を作ってやった。

6)　ラインを参考に，サークル，ボックス，そしてだんだんコピーだとかを作ってやる。拡大縮小，回転は面倒だから後回し。“うしろまわし”ではない。

7)　モード1は7割ほど完成。モード2はモード1をほとんどコピーしてやった。ただし領域枠，ルーペ用ズームなんかはがんばって作る。

8)　モード2もあらかたOK。余ってたところに女王様を描いておいてやろうと思って試しにSM.Xで絵を描く。セーブだとかはまだないから手作業でデータ化。

9)　面倒だったモード0を考える。指定方法なんかはいままででだいたいわかっていたから，カット&ペーストぐらいつけてやる。

10)　ファイル関係もそろそろやってやる。使用方法は適当にこのとき考えた。いい加減に作ってやる。

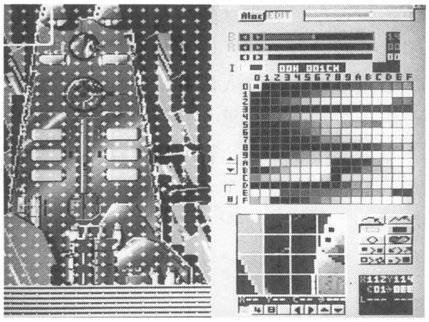
11)　だいたいできてきた。セーブなんかの指定がいまいちだった。スプライトさえも矩形領域しかセーブできなかったのだ。だから実際のヤツのやり方もプログラムしてやった。指定したものが判別しにくい。へボくて情けないが点滅にしてやる。ちょっとはわかりやすいからOKとした。

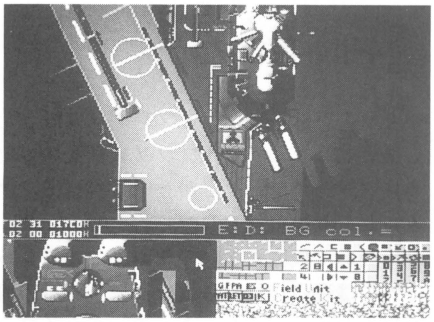
12)　テスト期間だからしばらく放っておく。テストが終わって久しぶりに修正していってやる。ソースの汚さに絶望する。だがなんとか完成させた。やはり面倒でもドキュメントは書くべきであったといつもどおり反省した。

13)　本来使えるルーチンをピックアップして，将来にそなえてやるのだがなんにせよソースが腐っている。見捨てた。

そして，皆さんには渡ることがないが私用の専門エディタを作った。F○CKってやつとECSTASYってやつだ。どういうエディタかというと，スクロールシューティングの背景ピクセルを作るField-Unit-Create-Kit，私のシステム用のマップエディタEdit&Create-STage-for-An-Shooting-sYstemだ。ひどいゴロ合わせだが構わない。

内容的にSM.Xと似ていることは想像つくことだろう。SM.Xのおかげでこれらはすぐ完成した。ひとつあたり3日でできたのだ。一時はもう二度と見ないだろうと思うモノでもこのように結局見ることがあるものだ。だから絶対，作ったモノの整理はしようではないか。


背景エディタ（参考）


マップエディタ（参考）

SM.Xの改良

本文とはあまり関係ありませんが，オマケとして6月号で発表したSM.Xの改良プログラムを掲載しておきましょう。リスト1のプログラムをSM.Sの4074行からつけ足してください。これにより，モード2時に“↓”でピックアップカラーがつきます。

●リスト1

```
KEYGT2:                                     *4073
          lea.l     ed2colors,a6            *from 4074
          move.w    #$7,d1                  *insert this source-program
          IOCS      _BITSNS
          btst.l    #6,d0
          beq       KEYGT2ON
          move.w    MSX,d0
          move.w    MSY,d1
          cmp.w     #122,d1
          bcs       KEYGT2ON
          cmp.w     #250,d1
          bcc       KEYGT2ON
          cmp.w     #120,d0
          bcs       KEYGT2ON
          cmp.w     #248,d0
          bcc       KEYGT2ON
          bsr       SUPER
          move.w    MSX,d0
          sub.w     #120,d0
          sub.w     #122,d1
          move.w    d0,d3
          move.w    d0,d4
          move.w    d1,d5
          move.w    d3,d0
          bsr       zoomgetxy
          add.w     ZMX,d0
          add.w     ZMY,d1
          swap.w    d1
          CLR.W     d1
          ror.l     #6,d1
          movea.l   d1,a0
          add.w     ED2X,d0
          add.w     d0,d0
          add.w     d0,a0
          adda.l    #$c00000,a0
          move.w    (a0),d0
          bsr       ed2colors
          lea.l     ed2colors,a6
*
```

FIN-反省

もう書いちゃったけどプログラムできたら反省しような。ライブラリに補充するためにフォーマットだとかを紙に書いてファイルしておいたりしながら。

一発ある程度大きいのをカマしたらきっと次のは楽に作れるに違いない。作業の流れはつかめているし，こうやってライブラリはたまるしな。IOCSのように，単純な作業をするサブルーチンは次回もほぼ変更なしに使い回せるのである。

だから巨大なものを作れば作るほど，いずれはガンガン楽にプログラムを作ることができるのだ。だからもう，ピックアップしまくってガンガン高速化，最適化，その他，やり込んでため込んでいこうではないか。

大きいプログラムを作るのは確かに難しいことだ。まずなにから手をつけてよいのかわからないし先は読めないし，その他，困難で満ちあふれているからだ。でもここまで読んでくれた人はちょいとは進め方がわかってくれたと思う。そのうちみんなできるようになるさ。管理能力はだんだん築かれていく。みんなヤンエグ野郎にいずれなれる。同時に技術もアップしていく。すれば独り歩きしていけるに違いない。

盗め！

私がかつてグラディウスを作ったときどうしたかを最後のテクニックとして公開しておこう。「解析」だ！　人様のプログラムをハックするのだ。技術を盗め，腕力を鍛えよ。強引だが実はこれがもっとも勉強となるのだ。

「盗むと泥棒じゃないか」などと不粋をいってはいけない。他人の作ったコードやデータを盗むのはいけないが，アイデアやノウハウは盗み盗まれるモノと相場が決まっている。カッコよくいうと「リバースエンジニアリング」というやつだ。ソフトウェアの技術なんてのは，みんな盗み盗まれて進化してきたのだ。だから，ソフトウェア技術の発展のためにも，臆せず堂々とやりなさい。

私はいまでもまだちょいとやっているし，やらなければいまこうしてプログラムを作ることなど不可能であった。

SION IIを盗め！　あれはソースが公表されている。なんたって変数名がわかっているのだから楽々盗れる。やりなさい。ぜひやりなさい。ソースを公開するってことは皆に盗めといっているということだ。よいチャンスなのだ。だってふつうの解析はまず逆アセンブル(DIS.Xなんかを使う)して，順に読んでいき，変数がなにを意味しているかを考え，厳しいアルゴリズムを理解していかなければならないのだから変数がわかっているのは素晴らしいことなのだ。

だからSM.Xでもソースプログラムを公開している(汚いので迷惑かもしれないが)。正直いって，本当なら人前にさらすのが恥ずかしいようなプログラムである。しかし，ちょっとした改造などをする場合，ソースがあるとないのとでは天と地の違いがある。それがどんなに醜いソースでもだ。

フリーソフトウェアなんかでも「ソースが汚い」ことを理由に公開しない人が多いようだが，デバッグや拡張を繰り返したプログラムがつぎはぎになるのは当たり前のことである。某氏曰く「動いているプログラムはすべて美しい」のだそうだ。

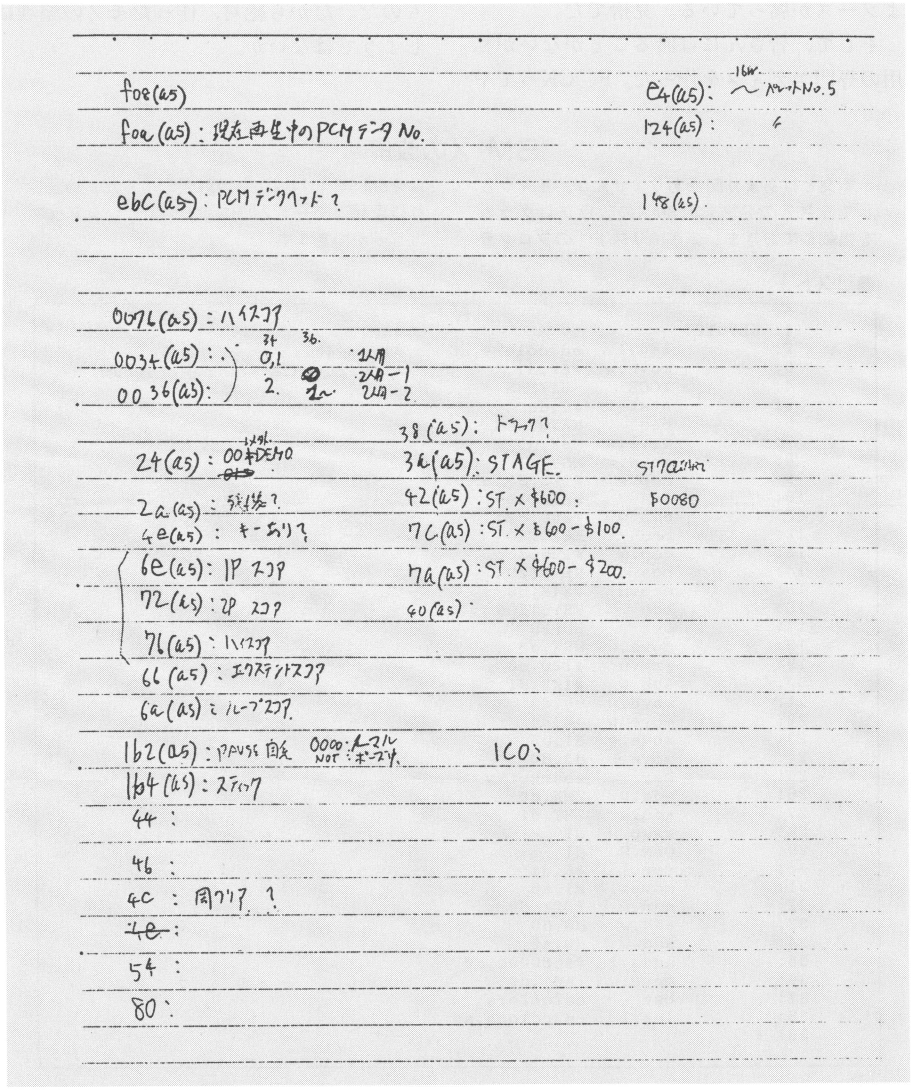
最後に

では，がんばってくれ。まとめると，まず“やりたい！”と思う野望の炎を燃やすことだ。これなしではいずれやる気がなくなってやめちゃうことになる。そしてうまい計画を立て，あれこれ考え，知力と腕力，さらに体力を駆使してバキバキと打ち込んでいくのだ。ナンパともあまり差はない，野望，力，強引さをフル活用して皆でX68000をヒイヒイいわせてやれ！

＊　＊　＊

ところでG2みんな燃えてるかな？　私の究極ラップはすでに21分02秒だ。計算上19分までは可能。でもそのためには宝クジ当てるぐらいの強運が必要だ。しかしそんな強運はぜひ宝クジに使おう。まだ上がいるかもしれないけど結構よいほうだと思っている。6-3が１分だって？　まだケツが青い。45秒までは伸びるものだ。がんばれ！

図1　横内君の解析メモ



WE WANT YOU!

Oh!Xの掲載記事を理解するうえで重要となるキーワードに「パーソナルコンピューティング」という言葉があります。なにも，難しい概念などではありません。Oh!Xが提唱しているのは，「パーソナルコンピュータをちゃんとパーソナルコンピュータとして使う」，というごく単純なことにすぎないのです。

それぞれの人がそれぞれのスタイルでパーソナルコンピューティングを楽しんでいると思います。それがどんなものであるかを知ることは，本誌の誌面作りにとって非常に重要なことなのです。そして，Oh!Xが発信したメッセージを皆さんが受け取り，それに対する皆さんのメッセージが今後のOh!Xの方向を決めていくことにもなります。

実際，Oh!Xの誌面はスタッフだけが作っているものではありません。これまでのOh!MZ/Xの軌跡をたどると要所要所で読者投稿作品が大きな影響力を及ぼしていることがわかります。読者の力がこれまでのOh!Xを支えていたといっても過言ではないでしょう。

しかし，影響を与えられているのは投稿作品だけではありません。実はそれ以上の影響力を持つのがアンケートハガキによるメッセージです。Oh!Xの全体的な方向性を決めているのは誌面にはあまり現れない多くの人の意見なのです。読者層が変われば記事が変わる，というほど単純なものでもありませんが，記事の方向性に多大の影響を及ぼしています。

投稿作品はそれ自体が強いメッセージでもあります。強いメッセージは歓迎します。また，アンケートハガキの回収にもご協力ください。多くの方の意見が揃ってこそ，よりよいフィードバックが行われます。

私たちはいつでも皆さんからのメッセージを求めています。

〈重点募集項目〉

- **オリジナル音楽データ**
- **カードゲーム**
- **SX-WINDOWアクセサリ**
- **Z's-EX用外部コマンド**

イラスト投稿の規定

サイズはハガキ大（A6判）以上であれば可。B5判くらいまでは可能ですが，取り扱いの手間や現実的な問題としてハガキ大を一応の標準とします。いずれにせよ，掲載時にはかなり縮小されることを考慮して描いてください。

一応の推奨形式は以下のとおりです。

1）　ハガキ大のケント紙で郵送

ハガキでも結構ですが，たまに裏面にも消印が押される場合があります。

2）　黒1色（薄ズミ不可）

墨汁は汚れの原因になることがあります。製図用インクがおすすめです。原稿は縮小されますのでスクリーントーンの80，90番台（レトラセットの場合）などや色の濃すぎるものについては再現は保証されません。残念ながら，カラー原稿はごくたまにしか掲載されません。

内容に関して特に規制はありませんが，時期もの（正月，クリスマス，季節もの）などについては，掲載が予想される時期を考慮して早めに送ったほうが有利になることがあります（年賀状は例外）。

それでは，皆さんの力作をお待ちしています。

協力スタッフ募集

Oh!Xでは誌面作りに参加していただく協力スタッフを募集しています。

スタッフとして活動する熱意があり，東京近郊にお住まいの方でソフトバンクまで来社可能な方。特に時間的な束縛はありませんが，ある程度時間的な余裕がある方に限ります。基本的に学生を対象としていますが，十分に時間的余裕と余力があれば社会人も可とします。ただし，18歳未満の学生および浪人生の方については採用予定はありません。

応募要項です。ライター希望の方はOh!X誌面2ページ分相当（2000字程度）の自由論文に自己紹介文を添えて「Oh!Xスタッフ希望」係までお送りください。

また，文章力には自信がないけどプログラムなら……という方でも技術スタッフとして，参加していただく場合があります。こちらを希望の方は自由論文の代わりに，これまでに制作した自作プログラムとその解説などを一緒に応募してください。

書類選考後，採用者の方にはこちらから連絡いたします。

投稿大募集

Oh!Xでは読者の皆さんによる投稿作品を常時募集しています。

未発表の作品であれば，グラフィック，音楽，システムプログラム，ツール，ゲーム，ハードウェアなどジャンルを問いません。数当てゲームからOSまでなんでも受け付けています。機種についても（メーカー，年代など）特に限定はしませんが，雑誌の性格上扱いにくい場合もあります。

誌面に載りきらない大きなアプリケーションなどはディスクメディアを使って配布することが考えられます。その形態のひとつはご存じ付録ディスク，そしてもうひとつは別冊形式によるものです（10月発売予定のZ-MUSICシステムに続き，今後もいくつかのOh!X MOOKシリーズが予定されています）。

また，特に掲載されることを目的とせず，「こんなものを作ってみました」といったプログラムでもかまいません。気軽に作品を送ってみませんか。

投稿募集要項

1）　お送りいただくプログラムには，住所，氏名，年齢，職業，連絡先電話番号，機種名，使用言語，動作に必要な周辺機器，マイコン歴などを明記のうえ，封書の宛先の最後には「Oh!X LIVE」，「全機種共通システム」，「投稿ゲームプログラム」など，プログラムの内容を明確にご記入ください。

2）　投稿されるプログラムには詳しい内容を記入した原稿を同梱してください。ディスクの中にドキュメントファイルの形式でのみ記述している方がいますが，郵送時の事故などでメディアが破壊されることもありますので，必ず文書を添えるようにしてください。一緒に変数表，メモリマップ，参考文献などがあればなお結構です。また，掲載に際してお送りいただいたプログラムやデータ原稿については，当方で加筆修正をさせていただくことがあります。

3）　お送りいただくプログラムは事故防止のため最低2回はセーブしておいてください。基本的に同封されたフロッピーディスク，カセットテープ，クイックディスク，原稿などについてはご返送いたしませんので，あらかじめご了承ください。

4）　ハード製作関係の投稿につきましては，最初は内容のわかる原稿のみお送りいただければ結構です。その後，当方で製作物が必要だと判断した場合には改めて連絡いたします。

5）　お送りいただいた作品の採用につきましては，掲載号が決定した時点で当方より連絡いたします。特にツール関係，ハード関係などのものにつきましては特集内容などを考慮したうえで採用決定されますので，結果を連絡するまでに時間がかかる場合があります。

6）　投稿いただいたプログラムにバグなどが発見された場合は新しいプログラムの入ったメディアと一緒に文書にてご連絡ください。

7）　掲載されたプログラムに対しては当社規定の原稿料をお支払いたします。また，投稿されたプログラムの著作権などはすべて制作者に保留されますが，いわゆる「PDSなどとしてネットにアップする」ことなどを希望される場合には必ず事前に編集部までご連絡ください。なお，一般的モラルとして，他誌との二重投稿または，他誌に掲載されたプログラムの移植などについては固くお断りいたします。

その他，不明点については編集室まで問い合わせてください。

宛先
〒108　東京都港区高輪2-19-13　NS高輪ビル
ソフトバンク株式会社
Oh!X編集室「投稿プログラム」係

バックナンバー案内

BACK ISSUES

ここには1991年8月号から1992年7月号までをご紹介しました。現在1991年1，5，9，11，12，1992年1～7月号の在庫がございます。バックナンバーおよび定期購読の申し込み方法については，161ページを参照してください。

1991

8月号（品切れ）
特集　印刷の世界へ

連載　大人のためのX68000/SX-WINDOW/ようこそC言語
響子 in CGわ～るど/ハード工作/ショートプロぱーてぃ
吾輩はX68000である/マシン語プログラミング

- X68000カードゲーム　七並べ
- X1用ゲーム　DEFEAT2

LIVE in '91　パワードリフト/イースIII/TURBO OUTRUN
THE SOFTOUCH　黄金の羅針盤/サイレントメビウス/パロディウスだ!他
全機種共通システム　Small-C ライブラリの移植

9月号
特集　Brush up your MAGIC.

連載　マシン語プログラミング/DōGA/Z80's Bar/ショートプロ
響子 in CGわ～るど/ハード工作/シミュレーション入門
吾輩はX68000である/大人のためのX68000/C言語

- X1用ゲーム Manual Runner
- ANOTHER CG WORLD

LIVE in '91　One/WHITE MANE
THE SOFTOUCH　イース/生中継68/アークス・オデッセイ他
全機種共通システム　SLANG用NEWファイル入出力ライブラリ

10月号（品切れ）
特集　マシン語との邂逅

連載　響子 in CGわ～るど/マシン語プログラミング/ショートプロ
ハード工作/Z80's Bar/よいこのSX-WINDOW/ANOTHER CG WORLD
吾輩はX68000である/ようこそC言語/大人のためのX68000

- 新連載　Computer Music入門
- NEW Print Shop PRO-68K Ver 2.0

LIVE in '91　うれしい!　たのしい!　大好き/SPANISH BLUE
THE SOFTOUCH　ボナンザブラザーズ/ロードス島戦記/ジーザスII他
全機種共通システム　Small-C活用講座（初級編）

11月号
特集　空間彷徨型ゲーム大分析

連載　響子 in CGわ～るど/大人のためのX68000/ANOTHER CG WORLD
DōGA/ショートプロ/Computer Music入門/吾輩はX68000である
ようこそC言語/マシン語プログラミング/Z80's Bar/ハード工作

- X68000用カードゲーム　ギャップ
- 新製品紹介　F-Card GT

LIVE in '91　オーダイン
THE SOFTOUCH　キャメルトライ/アクアレス/フューチャーウォーズ他
全機種共通システム　Small-C活用講座（応用編）/MORTAL

12月号
特集　音・そして音楽とコンピュータ
別冊付録　X68000 THE GAME SOFTWARE BEST SELECTION

連載　響子 in CGわ～るど/マシン語プログラミング/ショートプロ
ハード工作/Z80's Bar/ようこそC言語/ANOTHER CG WORLD
吾輩はX68000である/Computer Music入門/大人のためのX68000

- エレクトロニクスショウ＆データショウ

LIVE in '91　OH YEAH!/サイレント・イヴ/ジングルベル
THE SOFTOUCH　フェアリーランドストーリー/プロサッカー68他
全機種共通システム　Small-C用　SLANGコンパチ関数

1992

1月号
特集　SX-WINDOWの未来

連載　響子 in CGわ～るど/DōGA・CGA/大人のためのX68000
ハード工作/Z80's Bar/ショートプロ/吾輩はX68000である
ANOTHER CG WORLD/Computer Music入門/カードゲーム

- MAGIC用ゲーム　3DMAZE
- CM-300/500&LA音源の活用法

LIVE in '92　DRAGON SABER/すき/THE ENTRETAINER
THE SOFTOUCH　出たな!! ツインビー/ブリッツクリーク/飛翔鮫他
全機種共通システム　パズルゲームLINER

2月号
特集　2Dグラフィックの拡張

連載　響子 in CGわ～るど/大人のためのX68000/マシン語プログラミング
ハード工作/ショートプロ/ANOTHER CG WORLD/Z80's Bar
吾輩はX68000である/Computer Music入門/カードゲーム

- TREND ANALYSIS
- MIRAGE MODEL STUFF/Press Conductor PRO-68K

LIVE in '92　ストリートファイターII/Tide Over
THE SOFTOUCH　ジェノサイド2/アルシャーク/コード・ゼロ他
全機種共通システム　シミュレーションゲームPOLANYI

3月号
特集　SCSIの活用

連載　響子 in CGわ～るど/DōGA・CGA/大人のためのX68000/Z80's Bar
ショートプロ/吾輩はX68000である/マシン語プログラミング
ハード工作/ANOTHER CG WORLD/Computer Music入門/カードゲーム

- Z-MUSIC支援ツール　ZPDCON.X
- Z's-EX用拡張コマンド　MASK_reverse

LIVE in '92　ギャラクシーフォース/君が代
THE SOFTOUCH　グラディウスII/レミングス/大戦略/III'90/伊忍者
全機種共通システム　カードゲームKLONDIKE

4月号
特集　成熟するゲームと日本の文化

連載　よい子のSX-WINDOW/大人のためのX68000/Z80's Bar
響子 in CGわ～るど/ショートプロ/吾輩はX68000である
ハード工作/ANOTHER CG WORLD/Computer Music入門

- 発表　1991年度　GAME OF THE YEAR
- バーコードバトラー

LIVE in '92　あじさいのうた/ショパン練習曲作品25-2へ短調/It,s MAGIC
THE SOFTOUCH　ファーストクィーンII/マスターオブモンスターズII他
全機種共通システム　実践Small-C(1)オプティマイザ080

5月号
特集　明日のための環境づくり
第7回　言わせてくれなくちゃだワ

連載　響子 in CGわ～るど/大人のためのX68000/Z80's Bar
ハード工作/ショートプロ/マシン語プログラミング
Computer Music入門/吾輩はX68000である

- 製品紹介　MIDI音源　03R/W/MIC 68K

LIVE in '92　フレンズ/Danger Line
THE SOFTOUCH　エイリアンシンドローム/苦胃頭捕物帳他
全機種共通システム　実践Small-C(2)COMMAND.OBJ

6月号
特別企画　Oh!MZ,Oh!X10年間の歩み
特別付録　創刊10周年記念PRO-68K（5″2HD）

連載　響子 in CGわ～るど/大人のためのX68000/マシン語プログラミング
ハード工作/ショートプロ/ANOTHER CG WORLD/Z80's Bar
吾輩はX68000である/Computer Music入門

- 新製品紹介　Z'sSTAFF PRO-68K ver.3.0

LIVE in '92　Shake the Street/Ancient relics
THE SOFTOUCH　スピンディジーII/ロイヤルブラッド/ライフ&デス他
全機種共通システム　実践Small-C講座(3)COMMAND.OBJ2

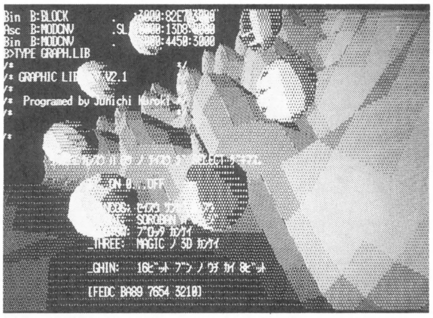
7月号
特集　超空間美術論
特別付録　DōGA CGAシステム&お試しディスク（5″2HD）

連載　よいこのSX-WINDOW/響子 in CGわ～るど/Z80's Bar
ANOTHER CG WORLD/大人のためのX68000
Computer Music入門/ハード工作/ショートプロ

- 試用レポート　V70アクセラレータボード

LIVE in '92　Bye Bye My Love/MATERIAL GIRL/ヴェクザシオン
THE SOFTOUCH　将棋聖天&棋太平68K/シムアース/太閤立志伝
全機種共通システム　実践Small-C講座(4)関数リファレンス

THE SENTINEL



〈対応機種一覧〉●MZ-80K/C/700/1500●MZ-80B/2000●MZ-2500/2861●X1●X1turbo/Z●PC-8001/8801/88●SMC-777/C●PASOPIA/5●PASOPIA 7●FM-7/77/AV●PC-286/386/9801/98●X68000
掲載されたプログラムの利用には各機種用のS-OS"SWORD"システムが必要です。

第122部　ワイルドカード　実践Small-C講座(5)

第123部　SLANG用MAGIC対応グラフィックライブラリ　GRAPH.LIB

●Small-C

Small-Cの再配布が終わりましたが，皆さんさっそく使っていますか。Small-Cを使ったアプリケーションの投稿は，まだ数が少ないのが実情です。皆さんの力作を心よりお待ちしています。

さて，今月の実践Small-C講座は，ワイルドカードに対応したdir，delコマンドの制作です。ワイルドカードは，特定の文字列で始まるファイルとか特定の拡張子をもったファイル，といったグループ形式でファイルを指定しそれを一気に削除したりできる便利な機能で，Human68kなどでお馴染みのものです。

用意されたワイルドカードは一般的な'*'と'?'です。前者は任意の文字列を，後者は任意の1文字を指定するのに使います。どのように使うのかは連載を参照していただくとして，ここでは「めったに使わないけれど，できると便利」なワイルドカードを紹介しておきましょう。それは，「ファイル名がABCで終わる（*ABC.*）」という形式の指定です。MS-DOSもHuman68kもこのパターンを指定することはできず，今回のSmall-C講座で作成したものも対応はしていません。いつかどこかで「どうしてないんだー」と叫び出してしまったら，ぜひとも投稿してください。

●グラフィックライブラリ

今月はもう1編。全機種共通システムの試みの中で誕生し，最近ではX68000のシューティングゲームSIONでも利用されているMAGICを，SLANGから使うためのライブラリが用意されました。

MAGICは表示したいデータを専用コードを使って書き下ろし，それを与えて「それ，描け」とやると描画を開始する，いわばバッチ処理的なシステムです。そのためデータを与えるのがなかなかに面倒な一面をもっているのですが，このライブラリでは描画命令の1つひとつを関数として実行できるようになっています。線を描きたいなら@LINE，ボックスを描きたいなら@BOXという関数を使えばいいというのですから，まさにBASIC感覚でプログラム中にMAGICを使って描かれたグラフィックを取り入れることが可能です。

単に描画関数を用意するだけでなくMAGICの描画時のワークエリアを操作するための配列なども用意されており，SLANGからMAGICを使いこなせるようになっています。実数計算にSOROBANを使い，MAGICの上にライブラリをかぶせてSLANGから使う。個々のモジュールが有機的に結合した，S-OSならではの一品といえるでしょう。SLANGで手軽に味わうことができるようになったMAGICの世界。ぜひ体験してみてください。

●S-OSの系譜(35)

1988年10月号では，奇しくもSLANG用の拡張ライブラリ第1弾が発表されています。今月のグラフィックライブラリに至るまで，SLANGは数々のライブラリによってその機能を拡張してきました。この10月号のライブラリは，1991年9月号で発表した「NEWファイル入出力ライブラリ」の先駆けとなったもので，最も基本的なファイル入出力関数であるファイルのオープン，クローズ，ファイル内の任意の場所へのシーク，そして1文字単位の入出力関数を提供しています。

このライブラリの登場でS-OSは，初めてファイル入出力の第一歩を踏み出したともいえるわけで，記念碑的な思い出深いライブラリです。SLANG用ライブラリという形をとってはいましたが，掲載されたマシン語プログラムは，SLANG以外のプログラムからも呼び出せるようになっていました。

またこの号には，シューティングゲーム「MANKAI」も発表されています。迫りくるミサイルを次々と迎撃していくミサイルコマンダーのようなゲームで，画面にミサイルの花が満開になる爽快感が魅力でした。S-OSの標準ルーチンが遅いと思われていたあの頃を振り返り，いま一度プレイしてみてはいかがでしょうか。

1992■インデックス

■92年1月号
第115部　LINER
■92年2月号
第116部　シミュレーションゲームPOLANYI
■92年3月号
第117部　カードゲームKLONDIKE
■92年4月号
第118部　オプティマイザO80実践Small-C講座(1)
■92年5月号
第119部　COMMAND.OBJ 実践Small-C講座(2)
■92年6月号
第120部　COMMAND.OBJ2実践Small-C講座(3)
■92年7月号
第121部　関数リファレンス実践Small-C講座(4)

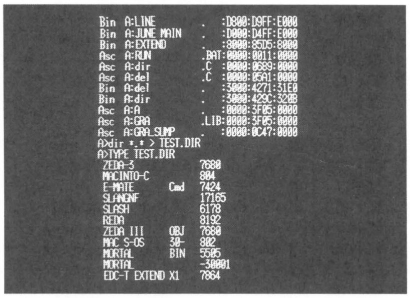
全機種共通
S-OS"SWORD"要

ワイルドカード

実践Small-C講座(5)

Ishigami Tatsuya
石上 達也

今回は，Small-Cの基本的な使い方を解説します。まだSmall-Cに慣れていない人は，ぜひ実践してみてください。サンプルとして，ワイルドカード対応の外部コマンドを2つ作成します。



ワイルドカード

今月は，COMMAND.OBJを発表したときに「誰か作ってくれないかな」といっていた，ワイルドカードを取り上げます。

皆さん知ってのとおりMS-DOSやHuman68kでは，このワイルドカードという概念がファイルの検索に使用されています。さらにエディタソフト(MicroEMACSなど)では，文字列の検索条件にまで使用されています。ワイルドカードを使えば，ファイルオペレーションをする場合に，拡張子が「．ASM」のものだけ表示しなさい，とか，名前が「TEXT」であれば拡張子の種類にかかわらず削除しなさい，などと命令することができるのです。

具体例を先にいうと拡張子が「．ASM」のファイルはすべて，

＊．ASM

という条件と合致します。拡張子がどうであれ，ファイル名がWZDであれば，

WZD．＊

と合致します。さらに，

WZD＊．＊

という条件は，WZD1．ASMやWZD2．RELやWZD.DEFやWZD.OBJなどのファイル名と合致します。

つまり"＊"は，すべての文字列と合致するジョーカーカードみたいな記号なのです(ほかのキャラクタよりも役割が飛び抜けているという意味でメタキャラクタという)。別のいい方をするとシステム内の文字列比較サブルーチンは，比較対象の文字列の中に"＊"の記号があった場合，そこから比較作業を放棄するのです。ということは，残念ながらワイルドカードを使おうと思ったら，"＊"をファイル名に使ってはいけなくなります。

そして，この"＊"記号と親戚関係にあるのが"？"記号です。これも同じようにメタキャラクタです。前者はどんな文字列に対しても一致するものですが，後者はどんな文字にも一致するものです。つまり，システム内の文字列比較ルーチンは"＊"を見つけたら，比較作業を放棄しますが，"？"を見つけた場合，該当文字の比較を放棄するだけで，次の文字から比較作業を再開します。

具体例を挙げると，

WZD?.ASM

と合致するファイル名は，WZD1．ASM，WZD2．ASM，WZDZ.ASM……などです。WZD35．ASMは合致しません。5文字目の"5"がよけいですから。この"？"もワイルドカードを使うときには，ファイルネームに使えなくなります。ご了承ください。

関数を考える

それでは，この機能をサポートする関数を作ってみましょう。ここで関数の名前をfilefindとし，以下の機能をもたせることにします。

●関数filefind(filename, dirno)

［引数］ char ＊filename;
int dirno;

［機能］ filenameで示されたファイルネームをカレントデバイス中から探してくる。なお，検索はディレクトリの先頭からdirno番目のファイルより開始する。

ここで，2番目の引数が必要な理由を説明していきます。以下のような状況を考えてみてください。ディスク内に，

test1．C
test4．ASM
testtest.C

という3つのファイルが存在するとします。そこに「test＊．C」の条件に合致するファイルを探すため，filefind関数を呼んだとします。すると，この関数は1番目の「test1．C」と返事をするでしょう。C言語で使われる関数は，ひとつの戻り値しかもてないことになっていますから，1回の呼び出しで合致するすべてのファイルを調べることができません。要するに，3番目の「testtest.C」を見つけても呼び出し側に知らせることができないのです。これでは，せっかくのワイルドカード機能がだいなしです。

これらの問題を解決するために2つ目の引数を使うのです。最初にdirnoに0を代入してfilefind関数を呼び出します。すると，「test1．C」を見つけた時点で，filefind関数は1を返します。次に，第2引数にこの1を入れて，もう一度filefind関数を呼んでやります。すると，検索は1番目のファイルより後ろから行われますので，今度は「testtest.C」が見つけられます。このとき，filefind関数はこのファイルがディスク内の3番目に位置していたということで，3を返します。

このような動作を繰り返し行っていけば，やがて，条件に合致するディスク内のすべてのファイルを見つけ出すことができるのです。もちろん，ディスク内にあるファイルの数には限りがあるわけで，いつかはディレクトリの最後まで検索が終了してしまうこともあります。そんなときにはfilefind

関数は－1を返しますので，返り値が－1になったらループを抜け出すようにするのです。

実際に作る

実際に作ったのがリスト1です。2行目から82行目までの，

```
#asm
  ⋮
  ⋮
#endasm
```

で囲まれた部分はインラインアセンブラといって，この部分に関してはコンパイラは何も行わず，そのままアセンブルファイルに転送します。そして，リスト1は見てのとおりほとんどがアセンブラで書かれています。関数をアセンブラで記述するには，C言語の引数をなんとかアセンブラで扱えるようにしなければなりません。Small-Cでは，

```
filefind(filename, dirno);
```

という関数呼び出しに対して，図1のように引数が積まれます（以前話したようにSmall-Cでは，ほかの標準的な処理系と式の値がスタックに積まれる順序が逆です。注意してください）。よって，これらの引数をZ80で呼び出すためには以下のようにします。

```
POP     BC
POP     HL      ;dirno
POP     DE      ;filename
PUSH    DE
PUSH    HL
PUSH    BC
```

これでHLレジスタにdirno, DEレジスタにfilenameが入ります。ただし，図1からもわかるように，これらの引数を取り出す過程で，どこかに関数の戻り番地を保存しなくてはいけません。とりあえず，BCレジスタにこの値を入れて保存することにしました。そしてDEレジスタには，ポインタ型の値でファイル名が収められたfilenameという，char型配列変数の始まるアドレスが入ります。

また，最初のほうで(_IBFAD)の最初のデータを5と比べて，これをはじいています。これはファイルの属性を見ているのですが，1はバイナリファイル，4はASCIIファイルでした。5は何かというと，WZDシリーズ以来私が勝手に使っている「ただいま書き込み中」の属性なのです。つまり，今回の関数では書き込み中のファイルを無視するようにしています。

サンプルプログラム

せっかくのC言語なのにほとんどアセンブラで組んでしまってはもったいない，というわけでリスト2です。世の中「つう」といえば「かあ」というように，ワイルドカードといえば出てくるのがディレクトリ表示です。使い方は簡単，COMMAND.OBJを使って拡張を行っている場合コマンドラインから，

```
A>dir A:*.ASM
```

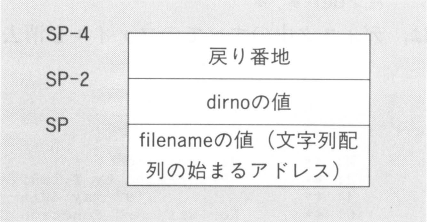
のように使用します。この場合は，Aドライブにある拡張子が“.ASM”であるファイルをすべて画面に表示します。そして，

```
A>dir *.*
```

はカレントドライブにあるすべてのファイルを表示します。

また，このプログラムは必ず小文字の“di

図1 スタックの構成



リスト1 filefind.C

```
 1: /*
 2: **       指定されたファイルネームを探す
 3: **                （ワイルドカード対応版）
 4: */
 5: filefind() {
 6: #asm
 7: _FCB   EQU 1FA9H
 8: _FILE  EQU 1FA3H
 9: _DIRNO EQU 1F67H
10: _IBFAD EQU 1F74H
11:
12:          POP     BC
13:          POP     HL          ;dirno
14:          POP     DE          ;filename
15:          PUSH    DE
16:          PUSH    HL
17:          PUSH    BC
18:
19:          LD      A,L
20:          LD      (_DIRNO),A
21:
22:          CALL    _FILE
23:          LD      HL,-1
24:          RET     C
25:
26:          LD      BC,32
27:          LD      DE,FnBuf
28:          LD      HL,(_IBFAD)
29:          LDIR
30:
31: File1:   CALL    _FCB
32:          JR      C,File2
33:          CALL    FNCMP
34:          JR      C,File1
35:
36:          LD      A,(_DIRNO)
37:          LD      H,0
38:          LD      L,A
39:          RET
40:
41: File2:   LD      HL,-1
42:          RET
43:
44: ;
45: ; File Name Compare
46: ; (Wild card support)
47: ;
48: FNCMP:   LD      HL,(_IBFAD)
49:          LD      A,(HL)
50:          CP      5          ;Being written
51:          JR      Z,Notmatch
52:          INC     HL
53:          LD      DE,FnBuf+1
54:          LD      B,16+1
55: FNCMP1:  DEC     B
56:          JR      Z,match
57:
58:          LD      A,(DE)
59:          INC     DE
60:          CP      (HL)
61:          INC     HL
62:          JR      Z,FNCMP1
63:          CP      '?'
64:          JR      Z,FNCMP1
65:          CP      '*'
66:          JR      NZ,Notmatch
67:
68:          LD      A,B
69:          CP      3+1
70:          JR      C,match
71:
72:          LD      HL,(_IBFAD)
73:          LD      DE,14
74:          ADD     HL,DE          ;HL=(_IBFAD)+14
75:
76:          LD      DE,FnBuf+14
77:          LD      B,3+1
78:          JR      FNCMP1
79:
80: match:   OR      A
81:          RET
82:
83: Notmatch:
84:          SCF
85:          RET
86:
87: FnBuf:   DS      32
88:
89: #endasm
90: }
```

r.C”で打ち込んでください。大文字の“DIR”とすると，COMMAND.OBJ自身の組み込み命令として処理されてしまいます。コンパイル後のオブジェクトファイルは，拡張子を除いた“dir”にしておくと便利でしょう。

さて，「つう」といえば，「与ひょう」というように（拍手），世の中「ワイルドカード」といえば，「ファイルの削除」です（リスト3）。

いままでは，いらないファイルをひとつずつDELコマンドで消去していきましたが，ワイルドカードがあればそんなチマチマした作業から解放されます。WZD1.ASM，WZD2.ASM，WZD35.ASM，WZD4.ASMなどというファイルを消したかったら，

A>del WZD＊.ASM

一発でOKです（でも，本当は消してほしくない石上）。ちなみに，

A>del ＊.＊

は，ディスク中のすべてのファイルを消去してしまいます。

このプログラムも“dir”コマンドと同様に，内部コマンドと区別するため小文字の“del”（拡張子はなし）としておいてください。

コンパイルの仕方

Small-Cというのは，ソース（原型）プログラムをファイルとして入力し，オブジェクト（目的）ファイルとして出力するプログラムです。したがって，SLANGやWZDのようにそのシステム内でプログラムを入力することはできません。ソースプログラムの入力は，エディタと呼ばれるプログラムを用いる必要があります。

さて，プログラムの入力は終わりましたか。リスト2はファイル名“dir.C”，リスト3はファイル名“del.C”でセーブします。そうそう，これら2つのファイルの後ろにリスト1の関数を追加するのを忘れずに。そしてファイルをセーブしたディスクに，配布ディスク中のSC，WZD,WLK,WLB,clib.LIBをコピーして入れてください。次にCOMMAND.OBJを立ち上げます。

このディスクをAドライブに入れます。カレントディスクはAになっていますか。どこがカレントかというのは，プロンプトでわかります。A>でない場合は，

B>A:

として，プロンプトをA>にしてください。

それでは，まず第1段階のC言語からアセンブラ言語への変換を行います。

A>SC dir −M −A −P −O

で，この作業が開始されるはずです。エラーが表示されたらもう一度エディタで，よくリストを見比べてください。めでたくエラーメッセージが表示されない場合は，コンパイル成功です。試しに，

A>DIR

と入力して，“dir.ASM”というファイルが作成されていることを確かめてください。アセンブラファイルが作成されたら，WZDを使ってアセンブルします。

リスト2 dir.C

```
/*
**        dir command
**         Programmed by T.Ishigami
**                  '92 May 30th
**        For Wildcard Function Sample
*/

main(argc, argv) int argc; char *argv[]; {
int dirno, cnt;
char *p;

 p = argv[1];
   if(argc != 2) p="*.*";
   if(strlen(p) == 2 && p[1] == ':') {
        p = "A:*.*";
        *p = *argv[1];   /* copy Device name */
        }

        cnt = 0;
        dirno = filefind(p,0);
        while(dirno != -1) {
                fprnt();
                cnt++;
                dirno = filefind(p, dirno);
        }
        if(cnt)
                printf("%d file(s)¥n",cnt);
        else
                printf("File not Found¥n");
}

fprnt() {
#asm
        LD      HL,(_IBFAD)
        INC     HL
        PUSH    HL      ;File name address
        LD      DE,11H
        ADD     HL,DE
        LD      A,(HL)
        INC     HL
        LD      H,(HL)
        LD      L,A
        PUSH    HL      ;File size
        CALL    putOut
        POP     BC
        POP     BC
#endasm
}

putOut(str,num) char *str; int num; {
        /*ファイルネームを表わす
        **文字列の終了記号を入れる */
        str[17] = 0;
        printf(" %s  %d¥n",str,num);
}
```

リスト3 del.C

```
/*
**        del command
**           Programmed by T.Ishigami
**                  '92 May 30th
**        For Wildcard Function Sample
*/

main(argc, argv) int argc; char *argv[]; {
  int  dirno, cnt;
  char *p;

    p = argv[1];
    if(argc != 2) p="*.*";
    if(strlen(p) == 2 && p[1] == ':') {
        p = "A:*.*";
        *p = *argv[1];      /* copy Device name */
    }

    cnt = 0;
    dirno = filefind(p,0);
    while(dirno != -1) {
        kill();
        cnt++;
        dirno = filefind(p, dirno);
    }
    if(cnt)
        printf("%d file(s)¥n",cnt);
    else
        printf("File not Found¥n");
}

kill() {
#asm
   CALL 1F9DH        ;FPRNT
   CALL 1FEBH        ;NL
   CALL 2015H        ;KILL
#endasm
}

```

A>WZD =dir

このWZDは，リロケータブルアセンブラといって，アセンブラファイルから，いきなりオブジェクトファイルを出力しません（ちなみに，いきなり出力するアセンブラは，アブソリュートアセンブラと呼ばれています)。リロケータブルファイルという中間形式のファイルを出力します。この中間形式のファイルをWLKを使って，コンピュータが実行できるような形のオブジェクトファイルに変換していきます。

A>WLK/P:3000,dir,clib/S,dir/P

で，コンピュータが直接実行できる形式のファイルになったはずです。もう一度，

A>DIR

と入力して“dir.OBJ”というファイルが出来上がっているか確認してください。出来上がっていたら，

A>REN dir.OBJ dir

で，拡張子を取りましょう。以上で，プログラム“dir.C”のコンパイル完了です。前述のように，このプログラムが動作するか試してみてください。

リスト3の“del.C”のコンパイルも同様に行います。キーボードから入力するコマンドを順に並べていくと以下のようになります。

```
A>SC dir −M −A −P −O
A>WZD =del
A>WLK /P:3000,dir,clib/S,del/P
A>REN del.OBJ dir
```

プログラムの説明

●dir.C

C言語で（ランタイムルーチン内のスタートアップルーチンを除いて）初めに実行されるのは，“main”という名前の関数であるという決まりがあります。このプログラムが起動されると，8行目からの関数mainが初めに実行されます。

関数mainの引数argc,argvには，起動されたときのコマンドラインに関する情報が入ります。argcには引数の数，argvには引数文字列のベクタテーブルのアドレスが入ります。

そして，12行目で1番目の引数のアドレスをポインタpに代入しています。たとえば，このプログラムが，

A>dir test.*

とコマンドラインから呼び出された場合，このポインタpの示すアドレスには“test.*”が収められています。

さて，引数の数がひとつでない場合。つまり，引数が2つ以上あったり，あるいはまったくなかった場合のことも考えなければなりません。それが13行目の，

if(argc != 2) p="*.*";

です。このargcというのは，自分のプログラム名（この場合だったらdir）も含めて，コマンドラインにいくつの文字列が並んでいたか，を示しています。よって，普通に考えて引数の数がひとつのときにはargc==2です。で，argcが2でないときには，カレントディスク内のすべてのファイルが表示されると解釈することにして，

p="*.*";

で，すべてのファイルが選択されるようにしておきます。

そしてもうひとつ，例外的な処理を行わなければなりません。

A>dir B:

以上のような入力に対する動作です。このようなときは，

A>dir B:*.*

と入力されたとみなして動作します。このような置き換えを行っているのが，14行から17行目までです。まず，14行目で引数の文字列が2文字からなっていることを確認し（詳しくは7月号の関数strlen()の項参照），その2文字目が“:”である場合で処理を振り分けています。

19行目で該当するファイル数をカウントする変数cntを初期化し，20行目で最初の該当するファイルを探しにいきます。このとき，該当するファイルが存在すれば（dirno≠−1），21〜25行のループに突入しますし，なければ26行目に飛びます。

ここのループでは，見つけたファイルの名前を表示し（関数fprnt），カウンタをひとつ増やしてから次の該当するファイルを見つけにいきます。次のファイルが見つからなければ（dirno=−1），ループを抜けます。見つければもう一度ループの先頭に戻ります。

ループを抜け終わった時点で，変数cntには表示したファイルネームの数が入っているはずです。これを26行で判別し，0でなければ27行でその数を画面に表示し，0であれば画面に“File not Found”と表示します。以上が関数main()の内容です。

そして関数fprntは，“SWORD”のサービスコール_FPRNT（現在扱っているファイルの名前を表示する）を呼び出して改行しているだけです。

●del.C

ほとんど，前述のdir.Cと中身は同じです。ただ，del.Cでは，検索条件に合致するファイルの名前を表示するのに加え，消去しなければなりません。それが，32行目からの関数kill()です。dir.Cに比べて，

CALL 2015H

の1行が追加されています。本来ならライブラリ関数unlink()を使用すればカッコいいのですが，これはASCIIファイルしか消去できない仕様になっています。ここでは仕方なく，インラインアセンブラで直接“SWORD”のサービスコール_KILLを呼び出すことにしました。

*　　*　　*

さて，次回はというと……うまくいけばちょっとしたプログラムの制作にとりかかることを予定しています。楽しみにしていてください。

Small-Cあれこれ

●リダイレクト機能について

今回のプログラムはSmall-Cで書いたわけですが，普通にマシン語で組むよりも大きくなっています。これはスタートアップルーチンがメモリを圧迫しているためです。その代わりとして，今回のプログラムではリダイレクト機能が使えます。

そこで，注意してほしいのは，リダイレクト機能を使った場合filefind関数に無視されることがあります。たとえば，

A>dir >test

のようにしてリダイレクト機能を使ったとします。すると，「test」というファイルには，ファイルの一覧表が出力されるのですが，この中に自分自身の名前「test」はありません。なぜなら，このファイルを書き込んでいるときは自分自身「書き込み中」なため，filefind関数に無視されているからです。

●Small-Cコンパイルオプション

次に本文中で使われていた，Small-Cのコンパイルオプションを説明します。

−A（Alarm）
コンパイル中にエラーが発生した場合，ビープ音を鳴らす。

−I（Initialize）
グローバル変数の初期値を0にする。

−M（Monitor）
現在コンパイラが対象としている関数名を画面に出力する。

−O（Optimize）
出力するコードを実行スピードを犠牲にして，プログラムサイズを抑えるようにする。

−P（Pause）
コンパイル中にエラーが発生した場合に，コンパイル作業を一時停止する。再開するためにはリターンキーを押す。

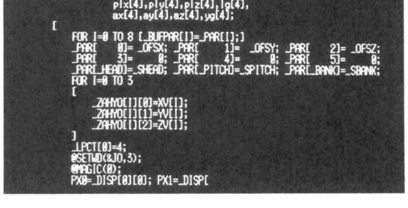
全機種共通
S-OS"SWORD"要

SLANG用MAGIC対応グラフィックライブラリ
GRAPH.LIB

要MAGIC, SOROBAN,SOROBAN.LIB

Kuroki Junichi
黒木 淳一

2D,3D機能のサポートにとどまらず，ポリゴン描画機能までサポートしたMAGIC対応ライブラリの登場です。このライブラリによって，もうひとつのMAGICの世界が見えることでしょう。



今回発表するプログラムは，SLANGでMAGICを簡単においしく利用するためのライブラリです。1990年9月号でも金子氏が，似たようなライブラリを発表されました。確かにそのプログラムを参考にしましたが，このライブラリはそれ以上に便利な機能を組み込んでいます。

機能としてはMAGICの機能をフルサポート，そのほかにもゲーム作成に便利な画面切り替え関数，ポリゴン描画関数などかなり充実しているでしょう。唯一の問題としてパレット機能を使う関数が，パレット機能を持たないMZ-2000/2200では正常に動きません。こればっかりはハードの制約ですので，がまんしてください。

メモリ節約の話

このライブラリは，ほかのソースプログラムにインクルードして使うことを前提としています。しかし，結構リストが大きいうえにSOROBAN.LIBまで組み込む必要があるため，ライブラリだけでもかなりメモリを消費します（すべての関数を利用しようとする人は諦めてください)。

そこで，このライブラリでは関数をいくつかに区切って，必要な関数だけをコンパイルすることができるようになっています。以下に説明する定数をライブラリを呼び出す，メインプログラムの先頭で定義することによって関数を切り離せるのです。定数の値が1のときにその関数を組み込み，0のときには組み込みません。

・_SINCOS

整数三角関数と整数逆三角関数，そしてこれらの関数を利用している関数の利用を選択します。

・_FLOAT

SOROBAN.LIBを使用している関数の利用を選択します。

・_PLOTSW

プロッタプリンタ出力関係の関数の利用を選択します。

・_THREE

3D関係の関数の利用を選択します。

・_TILESW

タイリングパターンを使用している関数の利用を選択します。

・_GHIN

上記の定数を利用してもまだメモリが足りない，または使わない関数が多いということがあります。そういうときは，この定数を利用して使用頻度の低い7つの関数を任意に削除することができます。消したい関数に対応するビットを0にしてコンパイルしてください。なお，上位バイトは意味を持ちません。

ビット	関数名
0	@SPLINE
1	@CPOLY, @KYU, @WAVE
2	@ROAD
3	@CFULL
4	reserve
5	@TLINE
6	@CLINE
7	@CBOX

具体的にどの関数が切り離されるかは，それぞれの関数中で説明されていますので，そちらを参照してください。

グローバル変数

次にGRAPH.LIBで使われているグローバル変数の説明をします。

●_CO [31]

この配列はMAGICのコマンド列を格納するためのものです。たとえば，LINE関数が呼び出されたときに，MAGICのLINEコマンドを実行するために必要なコマンドが，この配列に生成されます。ユーザーが使う必要はありません。

●_ZAHYO [P] [I]：$C6B6

空間図形の頂点座標を格納しています。

P＝0～255（頂点番号）
I＝0 頂点X座標
＝1 頂点Y座標
＝2 頂点Z座標

●_WIRE [P] [I]：$CCB6

空間図形の線分情報を格納しています。

P＝0～255（線分番号）
I＝0 頂点1
＝1 頂点2

●_LPCT [1]：$C6B4

_LPCT[0]には頂点の数，_LPCT[1]には線分の数が格納されます。

●_DISP [P] [I]：$C2B4 (READ ONLY)

空間図形の2D変換後の座標が格納されます。

P＝0～255（変換後の座標番号，_ZAHYOのPと対応）

I＝0　変換後のX座標

＝1　変換後のY座標

●_PAR［I］：$C203

3D→2D変換用のパラメータが格納されています。

I＝0：CX

＝1：CY　物体の位置

＝2：CZ　（オフセット）

＝3：DX

＝4：DX　回転の中心座標

＝5：DX

＝6：HEAD

＝7：PITCH　回転角度

＝8：BANK

上記の5つの配列は，MAGICの3D表示機能を使うときに利用されるものです。見てのとおり，これらの配列はMAGICの共通ワークエリアにかぶせて定義されているため，これらの配列への代入→MAGICへのデータ定義とすることもできます。当然，配列への定義用関数は用意されています（@SETOD(),@SETWD()参照）。

また，_PAR[]だけはユーザーの手で直接書き込んでやらなくてはなりません。そのほかの配列については，ライブラリ関数やMAGIC側で書き込んでくれます。そして，どの配列もユーザーの手で書き込むことができます。普通はこれらの配列をアクセスすることはあまりないはずですが，変則的な使い方をするとき，たとえば，アニメーション処理を複数パターンを使わず，配列に格納されている座標の変更だけで行うことができます。

●_TILE1, _TILE2

これらの変数は，MAGICのCIRCLE,TRIANGLE,BOX,FILLを描画するときに必要なタイルパターンを格納しておくためのもので，2つでひと組となっています。特殊な形でなくていいなら@GRAD関数でタイルパターンを生成してくれるため，ユーザーが直接触ることは少ないでしょう。

●_MSCREEN

この変数には@CRTKN関数を使用するときに，BLUE,REDのどのプレーンに書き込むかという情報を格納しています。

●_SPITCH　PITCH角度

●_SHEAD　HEAD角度

●_SBANK　BANK角度

●_OFSX　オフセットX座標

●_OFSY　オフセットY座標

●_OFSZ　オフセットZ座標

これらのパラメータは，ユーザーが定義したときに必要なパラメータ(_PAR[])とは違い，@TLINE(),@CPOLY(),@WAVE()などの，直接3Dグラフィック描画関数を扱う関数を使うときに利用されるパラメータです。

●_DEMODE

@CLINE(),@CBOX()の表示モードを格納します。0でPRESET，1でXORモード，2でORモードになります。

●_KSTEP

@WAVE(),@KYU()のステップ角度を格納します。デフォルトは15です。

●_SBLUE,_SRED,_SGREEN

@CPOLY(),@WAVE(),@KYU()のポリゴン描画する面の色の濃度を格納します。範囲は0～7までです。

●_LX,_LY,_LZ

@CPOLY(),@WAVE(),@KYU()実行時の光線ベクトルを格納します。

●_GMASK

@CPOLY,@CFULLが描画する3プレーンにマスクを指定します。_GMASKの下位3ビットがそれぞれのプレーンに対応していて，ビットが1のページをマスクします。ポリゴン関係の関数を使った場合全プレーンに影響するため，画面切り替えを使ったアニメーションが不可能です。そこでこのマスク機能を使って，うまく1プレーンにのみ描画するための機能です。

●WSEL

これは@CPOLYでの描画モードを指定するものです。指定は下位4ビットで行います。それぞれのビットを立てることにより機能します。

ビット	機能
0	ポリゴンとワイヤーフレームで描画します。
1	ポリゴンの表示をキャンセルします。
2	陰面処理付きのワイヤーフレーム表示にします。
3	ポリゴンを描画してからポリゴンの輪郭を緑色のワイヤーフレームで描画します。

関数の説明

では，ここからGRAPH.LIBで使用できる関数を説明していきます。

●@LINE(X1, Y1, X2, Y2)

(X1, Y1)－(X2, Y2)を結ぶ直線を単プレーンに書き込みます。

●@SPLINE(X1, Y1, X2, Y2, X3, Y3)

(X1, Y1),(X2, Y2),(X3, Y3)を結ぶスプライン曲線を単プレーンに書き込みます。_GHINの0ビットを0にしてコンパイルすると削除できます。

●@BOX(X1, Y1, X2, Y2)

(X1, Y1)を左上,(X2, Y2)を右下にしたボックスを単プレーンに書き込みます。

●@TRIANGLE(X1, Y1, X2, Y2, X3, Y3)

(X1, Y1),(X2, Y2),(X3, Y3)を結ぶ三角形を_TILE1,_TILE2で定義したタイルパターンで塗り潰します。_TILESWを0にしてコンパイルすると削除できます。

●@FULL(X1, Y1, X2, Y2)

(X1, Y1)を左上，(X2, Y2)を右下の座標にしたボックスを_TILE1,_TILE2で定義したタイルパターンで塗り潰します。_TILESWを0にしてコンパイルすると削除できます。

●@CIRCLE(X1, Y1, R1)

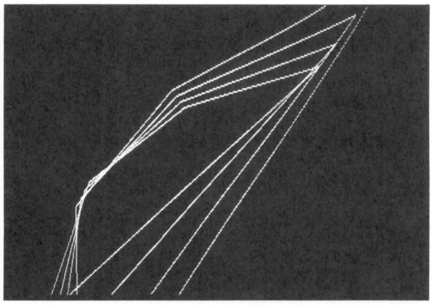
(X1, Y1)を中心とした半径R1の円を_TILE1,_TILE2で定義したタイルパターンで塗り潰します。_TILESWを0にしてコンパイルすると削除できます。

●@WINDOW(X1, Y1, X2, Y2)

グラフィック描画範囲を(X1, Y1)－(X2, Y2)にします。

●@MODE(MODE,PLANE)

単プレーン書き込み関数の書き込むプレーンの設定と，描画モードの設定をします。それぞれの引数の値は以下のとおりに対応しています。



ページ切り替えによってちらつきをなくす

```
      MODE    PLANE
  0：  PRESET  BLUE
  1：  XOR     RED
  2：  OR      GREEN
```

●@CLS()

単プレーンの画面消去を行います。

●@PALET(A0, A1, A2, A3, A4, A5, A6, A7)

パレットの変更を行います。

●@INIT()

GRAPH.LIBの初期化を行います。必ず最初に実行するようにしてください。

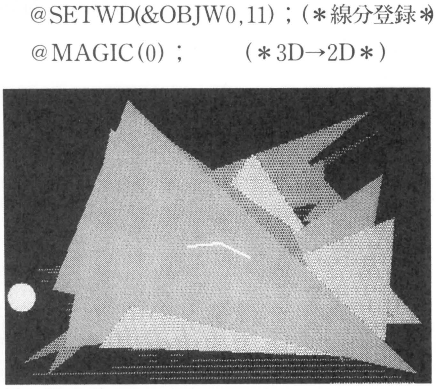
●@SETOD(POSIT,LENGTH)

●@SETWD(POSIT,LENGTH)

ユーザーが作成した空間図形をMAGICに登録します。@SETODは頂点座標の定義，@SETWDは線分の定義です。_THREEを0にしてコンパイルすると削除できます。

(例)

```
ARRAY WORD OBJD0[7][2]= [
        %-50,% 50,%-50,
        %-50,% 50,% 50,
        % 50,% 50,% 50,
        % 50,% 50,%-50,
        %-50,%-50,%-50,
        %-50,%-50,% 50,
        % 50,%-50,% 50,
        % 50,%-50,%-50 ];
    BYTE OBJW0[11][1]= [
        0,1, 1,2, 2,3, 3,0
        4,5, 5,6, 6,7, 7,4
        0,4, 1,5, 2,6, 3,7 ];
  MAIN()
   [
  @INIT();      (*初期化*)VAR I;
  @SETOD(&OBJD0,7) ;(*線分登録*)
    FOR I=O TO 8
    _PAR [I]=0;
  @SETWD(&OBJW0,11);(*線分登録*
  @MAGIC(0);        (*3D→2D*)
  @MAGIC(1);        (*表示*)
   ]
```



跳ね回るボールのデモ

●@MAGIC(COM)

COM=0のときMAGICに登録された空間図形を3D→2D変換します。COM=1のときは2D変換された空間図形を表示します。

●@CRTKN(COLOR,GR,PRW)

呼び出されるたびに描画ページをBLUE, REDの2ページを交互に切り替えます。画面表示を行ったあとに，この関数を実行してCLS()を使って画面消去することにより，完全に画面のちらつきをなくすことができます。また，余ったGREENプレーンはバックグラウンドとして使うことができます。COLORはキャラクタの色（ただし単色のみ)，GRはバックグラウンドの色，PRW=0でキャラクタをバックグラウンドの上に表示，PRW=1でバックグラウンドの下に表示します。

●@GRAD(GR)

GRの値(0～7)に従って8段階のタイルパターンを生成し，_TILE1,_TILE2に格納します。_TILESWを0にしてコンパイルすると削除できます。

●@ROAD(AA)

AAの値(0～7)に従った濃度で，画面の奥からグラデーションのバックグラウンドを描いていきます。

●@FLASH(C1, C2)

C1で示されるパレット番号をC2で示されるパレットコードに変えます。C1のパレットは保持されないことに注意。

●@PLINIT()

プロッタプリンタ用の関数を初期化します。動作確認をしたものはMYPLOTです。プリンタポートに接続してMML風にコマンドを送るものならたぶん大丈夫でしょう。_PLOTSWを0にしてコンパイルすると削除できます。

●@PLINE(X1, Y1, X2, Y2)

(X1, Y1)-(X2, Y2)を結ぶ直線をプロッタプリンタで印字します。_PLOTSWを0にしてコンパイルすると削除できます。

●@PLOT()

2D変換した空間図形をプロッタプリンタに出力します。PLOTSWを0にしてコンパイルすると削除できます。

●@TLINE(XS,YS,ZS,WE,YE,ZE)

(XS,YS,ZS)-(XE,YE,ZE)を結ぶ空間を直線で描画します。オフセットなどの座標パラメータは，_SHEAD,_SPITCH,_SBANK,_OFSX,_OFSY,_OFSZで与えてください。_THREE=0または_GHINの第5ビットを0にしてコンパイルすると削除できます。

●@CLINE(X1, Y1, X2, Y2, C)

(X1, Y1)-(X2, Y2)を結ぶ直線をカラーコードCで描画します。_GHINの第7ビットを0にしてコンパイルすると削除できます。

●@CFULL(X1, Y1, X2, Y2, C1, C2, C3)

(X1, Y1)を左上，(X2, Y2)を右下の座標にしたボックスをC1(BLUE),C2(RED),C3(GREEN)の濃度のタイルパターンで塗り潰します。指定の範囲は0～7までです。_TILESW=0または_GHINの第3ビットを0にしてコンパイルすると削除できます。

●@CPOLY(XV [], YV [], ZV [])

(XV [0], YV [0], ZV [0])
(XV [1], YV [1], ZV [1])
(XV [2], YV [2], ZV [2])
(XV [3], YV [3], ZV [3])

の4点を頂点とする面(ポリゴン)を描画します。_FLOAT=0でコンパイルするとワイヤーフレームで描画します。オフセットなどの座標パラメータは，_SHEAD,_SPITCH,_SBANK,_OFSX,_OFSY,_OFSZで与えてください。_THREE=0またはGHINの第1ビットを0にしてコンパイルすると削除できます。

(例)

```
  ARRAY WORD POLYX[3]= [
   %-50,%-50,% 50,% 50 ],
     WORD POLYY[3]= [
   %-50,% 50,% 50,%-50 ],
     WORD POLYZ[3]= [
   %000,%000,%000,%000 ];
  MAIN()
     [
     @INIT();
     _SBLUE=3;
     _SRED=3;
     _SGREEN=7;
     _OFSX=0;
     _OFSZ=100;
     _OFSY=0;
    @CPOLY(POLYX,POLYY,POLYZ);
   ]
```

●@KYU(MDX,MDY,MDZ,HNK)

(MDX,MDY,MDZ)を中心とした半径HNKの球体を描画します。オフセットなどの座標パラメータは，_SHEAD,_SPITCH,SBANK,_OFSX,_OFSY,_OFSZで与えてください。_THREE＝0または_SINCOS＝0または_GHINの第1ビットを0にしてコンパイルすると削除できます。

(例)

```
MAIN()
  [
   @INIT();
   _SBLUE=3;
   _SRED=3;
   _SBLUE=7;
   _OFSX=0;
   _OFSZ=0;
   _OFSY=0;
   @KYU(0,0,300,100);
  ]
```

●@ISIN(KAKU,HAN)

●@ICOS(KAKU,HAN)

半径(HAN) 0～32767に対する角度(KAKU) の三角関数の値を整数で求めます。SINCOSを0にしてコンパイルすると削除できます。

●@IATN(XX,YY)

XX,YYからアークタンジェントを求めます。使い道としては，ある物体が特定座標へ向こうとしたときの角度を算出できます。たとえば，AとBの2つの物体があったとします。Aの座標を（AX,AY），Bの座標を(BX,BY)として，R＝@IATN(BX-AX,BY-AY)とすればRにAから見たBへの角度を算出できるのです。_SINCOS＝0にしてコンパイルすると削除できます。

●WAVE(XSIZE,ZSIZE,POSY,SNP)

波を描画します。XSIZEはX方向の大きさ，ZSIZEはZ方向の大きさ，POSYはオフセットY座標，SNPは振幅となっています。オフセットなどの座標パラメータは，SHEAD,_SPITCH,_SBANK,_OFSX,_OFSY,_OFSZで与えてください。_TREE＝0かつ_SINCOS＝0かつ_GHINの第1ビットを0にしてコンパイルすると削除できます。

(例)

```
MAIN()
  [
   @INIT();
   _SBLUE=3;
   _SRED=3;
   _SBLUE=7;
   _OFSX=0;
   _OFSZ=0;
   _OFSY=0;
   @WAVE(300,300,0,50);
  ]
```

注）@FUKUGEN,@SCTLについては，単独で呼び出すことをしないでください。

ポリゴンについて

これらのライブラリ関数の中でポリゴンを描画する関数（@CPOLY）があります。これはただのポリゴンを描画するものではなく，光線ベクトルと法線ベクトルとの関係を計算して，表面の明るさ（色の濃度）を自動的に計算してくれます。アルゴリズムは個人的に代数の先生に教えてもらったり，先輩の家で夜の9時までかかって教えてもらったりしました。ちなみにフォンシェーディングというアルゴリズムを使っています。

ポリゴンを使うときの注意としては，SOROBAN.LIBをインクルードするため，コンパイルするときに_FLOAT＝1に設定してください。それほど精度は必要ないため，単精度5バイトで計算しています。

サンプルプログラム

どんなにプログラムを楽にするライブラリでも，サンプルがなければ使いこなすのに時間がかかるでしょう。ここでは，ひとおりGRAPH.LIBの機能を使用した，楽しい(？)デモを作りました。

このサンプルプログラムをコンパイルするためには，SOROBAN と SOROBAN.LIBが必要ですので用意しておいてください。SOROBANは9F00Ｈ番地からリロケートしておき，MAGICと一緒にセーブしておくと便利かもしれません。

エディタを使ってプログラムを入力して，とりあえずファイル名をTANOSHI.SLとします。コンパイル方法は，

```
] C TANOSHI.SL
```

と入力してください。コンパイルが終わると9000Ｈ番地からオブジェクトが生成されますので，

```
] S 3000 終了番地 3000 9000：TANOSHI
```

として，セーブしてください。

実行方法は，コマンドラインから，

```
#L SOROBAN
#L MAGIC
#L TANOSHI
#J3000
```

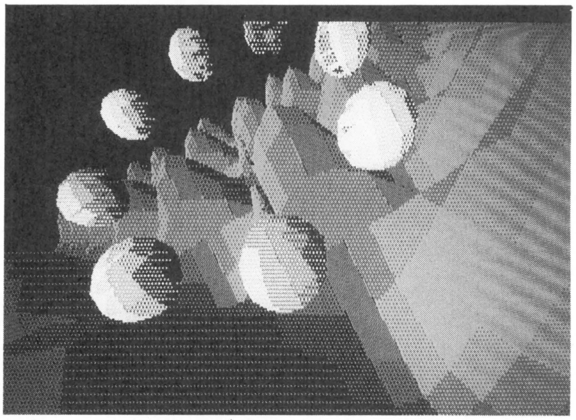
とすればOKです。

最後に

静止画から動画までこのライブラリ1本(とSOROBAN.LIB）で，なんでもできます。まあ，あまり欲張らなければですが。

また，こういったライブラリの制作は，マシン語を使うととっつきにくくデバッグにも時間がかかります。そんなとき，マシン語のようなスピードとリストの保守性のあるSLANGでMAGICを手軽に使えれば，と多くの人が思ったことでしょう。このライブラリがそういう人の役に立ち，X68000以上に8ビットマシンのMAGIC対応ソフトが増えてくれれば，嬉しいかぎりです。もちろん自分でもいくつかのアプリケーションを制作してあります。とりあえず，来月号でMAGIC用のモデラーを発表できると思います。楽しみにしていてください。

最後にライブラリを制作するにあたって，いろいろと注文してくれた深谷さん，ありがとうございました。



注目のポリゴン描画機能の表示例

リスト1

```
/*                                                             **
** SAMPLE PROGRAM 'TANOSI'  for 'GRAPH.LIB'                    **
**                                                             */

ORG     $3000;
OFFSET  $6000;

CONST    _PLOTSW=0,_THREE=1,_FLOAT=1,_SINCOS=1,_TILESW=1,_G
HIN=$FFFF;

#INCLUDE GRAPH.LIB

// MAGIC 3D DATA

CONST  PANEL_PMAX= 003,PANEL_WMAX= 003; (*座標、線分の数*)

ARRAY   PANEL_D[ 003][2]=[                (* 座標データ *)
          %-00100,%-00100,% 00000,        (* X , Y , Z *)
          %-00100,% 00100,% 00000,        (* X , Y , Z *)
          % 00100,% 00100,% 00000,        (* X , Y , Z *)
          % 00100,%-00100,% 00000],       (* X , Y , Z *)

 BYTE     PANEL_W[ 003][1]=[            (* 線分  データ *)
          000, 001, 001, 002, 002, 003, 003, 000];

VAR     CX,CY,X1,Y1,IX,IY,AX,AY,CC,AC,PR,KAKU,IKAKU,FL;

MAIN()
 VAR I;
 [
 @INIT();                             (*初期化          *)
 FOR I=0 TO 8 [ _PAR[I] = 0; ]        (*パラメータ  初期化  *)
 @SETOD( PANEL_D , PANEL_PMAX );      (*座標データ  定義    *)
 @SETWD( PANEL_W , PANEL_WMAX );      (*線分データ  定義    *)
 _PAR[2] = 5000;                      (*Z=5000         *)
 WHILE( _PAR[2] .>. 0 )             (*Zがマイナスになるまで*)
 [
    @MAGIC(0);                         (* 2D  →  3D      *)
    @MAGIC(1);                         (* 画面表示         *)
    @CRTKN( 7 , 0 , 0);                (* プレーン切り換え   *)
    @CLS();                            (* 裏プレーン消去     *)
    _PAR[2]=_PAR[2]-15;                (* Z=Z-15         *)
    _PAR[_HEAD]++;                     (*   HEAD回転      *)
    _PAR[_PITCH]=_PAR[_PITCH]+2;       (*   PITCH回転     *)
    _PAR[_BANK]=_PAR[_BANK]+3;         (*   BANK回転      *)
    IF (INKEY(0)==' ') EXIT;
 ]

   @INIT();
   CX = 319;              CY = 99;
   X1 = RND(640);         Y1 = RND(200);
   AX = RND(10) -  5;  AY = RND(10) -  5;
   CC = 0;                AC = 1;
   PR = 1;                FL = 0;
   _DEMODE=1;
   @MODE(2,2);
   FOR I = 0 TO 20
   [
       @GRAD(RND(5) + 1);
       @TRIANGLE(RND(640),RND(200),RND(640),RND(200),RND(6
40),RND(200));
   ]
   @MODE(2,0);
   WHILE(INKEY(0) == 0)
   [
     AX = AX + SGN(CX - X1);
     AY = AY + SGN(CY - Y1);
     IX = X1;
     IY = Y1;
     X1 = X1 + AX;
     Y1 = Y1 + AY;
     IF (X1 .<. 0)   [ X1 =  20;  AX =-AY; ]
     IF (X1 .>. 639) [ X1 = 619;  AX =-AY; ]
     IF (Y1 .<. 0)   [ Y1 =  10;  AY =-AY; ]
     IF (Y1 .>. 199) [ Y1 = 189;  AY =-AY; ]
     CC = CC + AC;
     IF (CC == 0) OR (CC == 7) [ AC = 0-AC; PR = 1-PR; ]
     @GRAD(CC);
     @CIRCLE(X1,Y1,20);
     IKAKU=KAKU;
     KAKU=@IATN(X1-CX,Y1-CY);
     @LINE(CX,CY,CX+@ICOS(KAKU,50),CY+@ISIN(KAKU,25));
     @CRTKN(7,1,PR);
     @GRAD(0);
     @CIRCLE(IX,IY,20);
     IF (FL==1) @LINE(CX,CY,CX+@ICOS(IKAKU,50),CY+@ISIN(IK
AKU,25));
     FL=1;
   ]
   @INIT();
   @MODE(2,2);
   _SPITCH=-30; _SHEAD=0; _SBANK=30;
   _KSTEP=30; _WSEL=0;
   _SBLUE=0; _SRED=7; _SGREEN=0;
   _OFSY=100;
   @WAVE(900,1200,60,130);
   I=0;
   WHILE(I<360)
   [
     _SBLUE=RND(4)+4; _SRED=RND(4)+4; _SGREEN=RND(4)+4;
     @KYU(@ICOS(I,500),-40-@ISIN(1,500)./.2,@ISIN(I,500)+9
00,100);
     I=I+45;
   ]
   ]
```

リスト2

```
/*                                */
/* GRAPHIC LIBRARY V2.1           */
/*                                */
/*  Programed by Junichi Kuroki   */
/*                                */

/*

          ツカワナイ カンスウ ハ 6ツ ノ テイスウ デ SELECT デキマス。

               1...ON 0...OFF

               _SINCOS: セイスウ サンカク カンスウ
               _FLOAT:  SOROBAN パッケージ
               _PLOTSW: プロッタ カンケイ
               _THREE:  MAGIC ノ 3D カンケイ

               _GHIN:   16ビット ブン ノ ウチ カイ 8ビット

               [FEDC BA89 7654 3210]
                |||| |||| |||| |||+--- : @SPLINE
                |||| |||| |||| ||+---- : @CPOLY
                |||| |||| |||| |+----- : @ROAD
                |||| |||| |||| +------ : @CFULL
                |||| |||| ||||
                |||| |||| |||+-------- : --------
                |||| |||| ||+--------- : @TLINE
                |||| |||| |+---------- : @CLINE
                |||| |||| +----------- : @CBOX
                |||| ||||
                |||| |||+------------- : --------
                |||| ||+-------------- : --------
                |||| |+--------------- : --------
                |||| +---------------- : --------
                ||||
                |||+------------------ : --------
                ||+------------------- : --------
                |+-------------------- : --------
                +--------------------- : --------

*/

CONST    _HEAD=6,_PITCH=7,_BANK=8;

#IF (_FLOAT==1)

#INCLUDE SOROBAN.LIB

#ENDIF

ARRAY    BYTE             _CO[31],
         WORD     _ZAHYO[255][2]:$C6B6,
         BYTE      _WIRE[255][1]:$CCB6,
         BYTE            _LPCT[1]:$C6B4,
         WORD              _PAL[7],
         WORD      _DISP[255][1]:$C2B4,
         WORD              _PAR[8]:$C203,
         WORD           _BUFPAR[8];

#IF (_TILESW==1)

ARRAY    WORD          _ROADY[7][1]=[
                 %00,%31,%32,%53,%54,%68,%69,%78,
                 %79,%86,%87,%91,%92,%96,%97,%99],
         WORD          _GRDTN[15]=[
                 $00,$00,$00,$00,$88,$00,$00,$00,
                 $88,$00,$22,$00,$88,$55,$22,$55,
                 $AA,$55,$AA,$55,$77,$AA,$DD,$AA,
                 $77,$FF,$DD,$FF,$FF,$FF,$FF,$FF];

#ENDIF

#IF (_SINCOS==1)

ARRAY    WORD     _STABLE[89]=[
    $00,$00, $3B,$02, $77,$04, $B2,$06, $ED,$08, $27,$0B,
    $61,$0D, $99,$0F, $D0,$11, $05,$14, $39,$16, $6C,$18,
    $9C,$1A, $CA,$1C, $F7,$1E, $20,$21, $47,$23, $6C,$25,
    $8D,$27, $AB,$29, $C6,$2B, $DE,$2D, $F2,$2F, $03,$32,
    $0F,$34, $17,$36, $1C,$38, $1B,$3A, $17,$3C, $0D,$3E,
    $FF,$3F, $EC,$41, $D3,$43, $B6,$45, $93,$47, $6A,$49,
    $3B,$4B, $07,$4D, $CD,$4E, $8C,$50, $46,$52, $F9,$53,
    $A5,$55, $4B,$57, $E9,$58, $81,$5A, $12,$5C, $9C,$5D,
    $1E,$5F, $99,$60, $0C,$62, $78,$63, $DC,$64, $38,$66,
```


▶ハミダシもSTUDIO Xだったんですね。「あの頃は若かった」というハガキが何十枚もくるでしょう。ちと恥ずかしい文が載ってしまったな。　橘　正彦(19)茨城県

```
    $8D,$67, $D9,$68, $1D,$6A, $58,$6B, $8B,$6C, $B6,$6D,
    $D9,$6E, $F2,$6F, $03,$71, $0B,$72, $0A,$73, $00,$74,
    $EE,$74, $D2,$75, $AD,$76, $7E,$77, $46,$78, $05,$79,
    $BB,$79, $67,$7A, $09,$7B, $A2,$7B, $31,$7C, $B7,$7C,
    $32,$7D, $A4,$7D, $0D,$7E, $6B,$7E, $C0,$7E, $0A,$7F,
    $4B,$7F, $82,$7F, $AF,$7F, $D2,$7F, $EB,$7F, $FA,$7F],
          WORD    _ATNTBL[63]=[
                   %00, %01, %02, %03, %04, %04, %05, %06,
                   %07, %08, %09, %10, %11, %11, %12, %13,
                   %14, %15, %16, %17, %17, %18, %19, %20,
                   %21, %21, %22, %23, %24, %24, %25, %26,
                   %27, %27, %28, %29, %29, %30, %31, %31,
                   %32, %33, %33, %34, %35, %35, %36, %36,
                   %37, %37, %38, %39, %39, %40, %40, %41,
                   %41, %42, %42, %43, %43, %44, %44, %45];
#ENDIF
VAR     _TILE1=65535,_TILE2=65535,_MSCREEN=0,
        _SPITCH=0,_SHEAD=0,_SBANK=0,_OFSX=0,
        _OFSY=0,_OFSZ=0,_BSWITCH,
        _OBFAD,_WBFAD,_OBUFF,_WBUFF,_DEMODE=2,
        _KSTEP=15,_SBLUE=7,_SRED=4,
        _SGREEN=4,_LX=30,_LY=10,_LZ=10,_WSEL,_GMASK=7;
@LINE(X1,Y1,X2,Y2)
  BEGIN
    _CO[0]=0;
    _CO[1]=2;
    MEMW[&_CO+2 ]=X1;
    MEMW[&_CO+4 ]=Y1;
    MEMW[&_CO+6 ]=X2;
    MEMW[&_CO+8 ]=Y2;
    _CO[10]=15;
    ^IX=&_CO;
    CALL($B004);
  END;
#IF ((_GHIN AND 1)==1)
@SPLINE(X1,Y1,X2,Y2,X3,Y3)
  BEGIN
    _CO[0]=1;
    MEMW[&_CO+1 ]=X1;
    MEMW[&_CO+3 ]=Y1;
    MEMW[&_CO+5 ]=X2;
    MEMW[&_CO+7 ]=Y2;
    MEMW[&_CO+9 ]=X3;
    MEMW[&_CO+11]=Y3;
    _CO[13]=15;
    ^IX=&_CO;
    CALL($B004);
  END;
#ENDIF
@BOX(X1,Y1,X2,Y2)
  BEGIN
    _CO[0]=2;
    MEMW[&_CO+1 ]=X1;
    MEMW[&_CO+3 ]=Y1;
    MEMW[&_CO+5 ]=X2;
    MEMW[&_CO+7 ]=Y2;
    _CO[9]=15;
    ^IX=&_CO;
    CALL($B004);
  END;
#IF (_TILESW==1)
@TRIANGLE(X1,Y1,X2,Y2,X3,Y3)
  BEGIN
    _CO[0]=3;
    MEMW[&_CO+1 ]=_TILE1;
    MEMW[&_CO+3 ]=_TILE2;
    MEMW[&_CO+5 ]=X1;
    MEMW[&_CO+7 ]=Y1;
    MEMW[&_CO+9 ]=X2;
    MEMW[&_CO+11]=Y2;
    MEMW[&_CO+13]=X3;
    MEMW[&_CO+15]=Y3;
    _CO[17]=15;
    ^IX=&_CO;
    CALL($B004);
  END;
#ENDIF
#IF (_TILESW==1)
@FULL(X1,Y1,X2,Y2)
  BEGIN
    _CO[0]=4;
    MEMW[&_CO+1 ]=_TILE1;
    MEMW[&_CO+3 ]=_TILE2;
    MEMW[&_CO+5 ]=X1;
    MEMW[&_CO+7 ]=Y1;
    MEMW[&_CO+9 ]=X2;
    MEMW[&_CO+11]=Y2;
    _CO[13]=15;
    ^IX=&_CO;
    CALL($B004);
  END;
#ENDIF
#IF (_TILESW==1)
@CIRCLE(X1,Y1,R1)
  BEGIN
    _CO[0]=5;
    MEMW[&_CO+1 ]=_TILE1;
    MEMW[&_CO+3 ]=_TILE2;
    MEMW[&_CO+5 ]=X1;
    MEMW[&_CO+7 ]=Y1;
    MEMW[&_CO+9 ]=R1;
    _CO[11]=15;
    ^IX=&_CO;
    CALL($B004);
  END;
#ENDIF
@WINDOW(X1,Y1,X2,Y2)
  BEGIN
    _CO[0]=6;
    MEMW[&_CO+1 ]=X1;
    MEMW[&_CO+3 ]=Y1;
    MEMW[&_CO+5 ]=X2;
    MEMW[&_CO+7 ]=Y2;
    _CO[9]=15;
    ^IX=&_CO;
    CALL($B004);
  END;
@MODE(MODE,PLANE)
  BEGIN
    _CO[0]=7;
    _CO[1]=MODE;
    _CO[2]=PLANE;
    _CO[3]=15;
    ^IX=&_CO;
    CALL($B004);
  END;
@CLS()
  BEGIN
    _CO[0]=9;
    _CO[1]=15;
    ^IX=&_CO;
    CALL($B004);
  END;
@PALET(A0,A1,A2,A3,A4,A5,A6,A7)
  BEGIN
    _CO[0]=10;  _PAL[0]=A0;
    _CO[1]=A0;  _PAL[1]=A1;
    _CO[2]=A1;  _PAL[2]=A2;
    _CO[3]=A2;  _PAL[3]=A3;
    _CO[4]=A3;  _PAL[4]=A4;
    _CO[5]=A4;  _PAL[5]=A5;
    _CO[6]=A5;  _PAL[6]=A6;
    _CO[7]=A6;  _PAL[7]=A7;
    _CO[8]=A7;
    _CO[9]=15;
    ^IX=&_CO;
    CALL($B004);
  END;
@INIT()
  BEGIN
    @WINDOW(0,0,639,199);
    @MODE(2,2); @CLS();
    @MODE(2,1); @CLS();
    @MODE(2,0); @CLS();
#IF (_TILESW==1)
    @GRAD(7);
#ENDIF
#IF (_THREE==1)
    _WSEL=1;
#ENDIF
    _MSCREEN=0;
    _BSWITCH=0; _DEMODE=2;
    @PALET(0,1,2,3,4,5,6,7);
    PRINT("¥C");
#IF (_FLOAT==1)
    @single();
#ENDIF
  END;
#IF (_THREE==1)
@SETOD(POSIT,LENGTH)
  VAR   I,K;
  BEGIN
    FOR I=0 TO LENGTH [
      FOR K=0 TO 2 [
        _ZAHYO[I][K]=MEMW[POSIT+I*6+K*2];
      ]
    ]
     LPCT[0]=LENGTH+1;
```

▶７月号65ページの写真を切り取り，適当な本のページの片隅に１枚ずつ張り付けますと，ちょっとしたパラパラマンガの出来上がり。さあ，あの「TORNADO」の感動がいまここに甦る……なんてね。　松本　拓司(18)埼玉県

```
  END;

@SETWD(POSIT,LENGTH)
  VAR   I,K;
  BEGIN
    FOR I=0 TO LENGTH [
      FOR K=0 TO 1 [
        _WIRE[I][K]=MEM[POSIT+I*2+K];
      ]
    ]
    _LPCT[1]=LENGTH+1;
  END;

@MAGIC(COM)
  BEGIN
    _CO[0]=13+COM;
    _CO[1]=15;
    ^IX=&_CO;
    CALL($B004);
  END;

#ENDIF

@CRTKN(COLOR,GR,PRW)
  VAR   A,B,C,D;
  BEGIN
    @MODE(_DEMODE,_MSCREEN);
    IF (_MSCREEN==0) [A=0; B=COLOR; C=GR; D=COLOR;]
    IF (_MSCREEN==1) [A=COLOR; B=0; C=COLOR; D=GR;]
    _MSCREEN=(_MSCREEN XOR 1);
    IF (PRW==0) [@PALET(0,A,B,A+B,GR,C,D,A+B);]
    IF (PRW==1) [@PALET(0,A,B,A+B,GR,GR,GR,GR);]
  END;

#IF (_TILESW==1)

@GRAD(GR)
  VAR   I;
  BEGIN
    I=GR*2;
    _TILE1=_GRDTN[I  ];
    _TILE2=_GRDTN[I+1];
  END;

#ENDIF

#IF (_TILESW==1) AND ((_GHIN AND 4)==4)

@ROAD(AA)
  VAR   I,J;
  BEGIN
    AA=AA AND 7;
    @GRAD(7-AA);
    I=_ROADY[AA][0]; J=_ROADY[AA][1];
    @FULL(0,I,639,J);
    @FULL(0,199-I,639,199-J);
  END;

#ENDIF

@FLASH(C1,C2)
  BEGIN
    _CO[0]=10;
    _CO[   1]=_PAL[0];
    _CO[   2]=_PAL[1];
    _CO[   3]=_PAL[2];
    _CO[   4]=_PAL[3];
    _CO[   5]=_PAL[4];
    _CO[   6]=_PAL[5];
    _CO[   7]=_PAL[6];
    _CO[   8]=_PAL[7];
    _CO[C1+1]=MEM[&C2];
    _CO[9   ]=15;
    _PAL[C1 ]=C2;
    ^IX=&_CO;
    CALL($B004);
  END;

#IF (_PLOTSW==1)

@PLINIT()
  BEGIN
    PRMODE(2);
    PRINT("H"); PRINT("¥N");
    PRINT("J1"); PRINT("¥N");
    PRMODE(0);
  END;

@PLINE(X1,Y1,X2,Y2)
  BEGIN
    PRMODE(2);
    PRINT("M "); PRINT((X1-319)*5+2000);
    PRINT(","); PRINT(1000-(Y1-99)*10); PRINT("¥N");
    PRINT("D "); PRINT((X2-319)*5+2000);
    PRINT(","); PRINT(1000-(Y2-99)*10); PRINT("¥N");
    PRMODE(0);
  END;

@PLOT()
  VAR   I,J,K;
  BEGIN
    @PLINIT();
    FOR I=0 TO _LPCT[1]-1 [
      J=_WIRE[I][0]; K=_WIRE[I][1];
      @PLINE(_DISP[J][0],_DISP[J][1],_DISP[K][0],_DISP[K][1]);
      ]
  END;

#ENDIF

#IF (_THREE==1) AND ((_GHIN AND 32)==32)

@TLINE(XS,YS,ZS,XE,YE,ZE)
  VAR   I;
  BEGIN
    FOR I=0 TO 8 [_BUFPAR[I]=_PAR[I];]
    MEMW[&_ZAHYO  ]=XS; MEMW[&_ZAHYO+2]=YS; MEMW[&_ZAHYO+4]=ZS;
    MEMW[&_ZAHYO+6]=XE; MEMW[&_ZAHYO+8]=YE; MEMW[&_ZAHYO+10]=ZE;
    _WIRE[0][0]=0; _WIRE[0][1]=1;
    _LPCT[0]=2; _LPCT[1]=1;
    _PAR[0]=_OFSX;        _PAR[1]=_OFSY;          _PAR[2]=_OFSZ;
    _PAR[3]=0;            _PAR[4]=0;              _PAR[5]=0;
    _PAR[_HEAD]=_SHEAD; _PAR[_PITCH]=_SPITCH; _PAR[_BANK]=_SBANK;
    @MAGIC(0);            @MAGIC(1);
    IF (_BSWITCH==1) [@FUKUGEN();]
    FOR I=0 TO 8 [_PAR[I]=_BUFPAR[I];]
  END;

@FUKUGEN()
  VAR   I;
  BEGIN
    @SETOD(_OBFAD,1);
    @SETWD(_WBFAD,0);
    _LPCT[0]=_OBUFF+1;
    _LPCT[1]=_WBUFF+1;
    IF (_BSWITCH==1) [FOR I=0 TO 8 [_PAR[I]=_BUFPAR[I];]]
  END;

#ENDIF

#IF ((_GHIN AND 64)==64)

@CLINE(X1,Y1,X2,Y2,C)
  BEGIN
    IF ((C AND 1)==1) [ @MODE(_DEMODE,0);   @LINE(X1,Y1,X2,Y2); ]
                ELSE  [ @MODE(2-_DEMODE,0); @LINE(X1,Y1,X2,Y2); ]
    IF ((C AND 2)==2) [ @MODE(_DEMODE,1);   @LINE(X1,Y1,X2,Y2); ]
                ELSE  [ @MODE(2-_DEMODE,1); @LINE(X1,Y1,X2,Y2); ]
    IF ((C AND 4)==4) [ @MODE(_DEMODE,2);   @LINE(X1,Y1,X2,Y2); ]
                ELSE  [ @MODE(2-_DEMODE,2); @LINE(X1,Y1,X2,Y2); ]
  END;

#ENDIF

#IF ((_GHIN AND 128)==128)

@CBOX(X1,Y1,X2,Y2,C)
  BEGIN
    IF ((C AND 1)==1) [ @MODE(_DEMODE,0);   @BOX(X1,Y1,X2,Y2); ]
                ELSE  [ @MODE(2-_DEMODE,0); @BOX(X1,Y1,X2,Y2); ]
    IF ((C AND 2)==2) [ @MODE(_DEMODE,1);   @BOX(X1,Y1,X2,Y2); ]
                ELSE  [ @MODE(2-_DEMODE,1); @BOX(X1,Y1,X2,Y2); ]
    IF ((C AND 4)==4) [ @MODE(_DEMODE,2);   @BOX(X1,Y1,X2,Y2); ]
                ELSE  [ @MODE(2-_DEMODE,2); @BOX(X1,Y1,X2,Y2); ]
  END;

#ENDIF

#IF (_TILESW==1) AND ((_GHIN AND 8)==8)

@CFULL(X1,Y1,X2,Y2,C1,C2,C3)
  BEGIN
    IF ((_GMASK AND 1)==1) [
                  @MODE(2,0); @GRAD(C1); @FULL(X1,Y1,X2,Y2); ]
    IF ((_GMASK AND 2)==2) [
                  @MODE(2,1); @GRAD(C2); @FULL(X1,Y1,X2,Y2); ]
    IF ((_GMASK AND 4)==4) [
                  @MODE(2,2); @GRAD(C3); @FULL(X1,Y1,X2,Y2); ]
  END;

#ENDIF

#IF (_THREE==1) AND ((_GHIN AND 2)==2)

@CPOLY( XV[], YV[] , ZV[] )
    VAR I,NX,NY,NZ,PX0,PX1,PX2,PX3,PY0,PY1,PY2,PY3,LV;
    ARRAY BYTE  JO[3][1]=[ 0,1, 1,2, 2,3, 3,0 ],
                plx[4],ply[4],plz[4],lg[4],
                ax[4],ay[4],az[4],yg[4];
    [
        FOR I=0 TO 8 [_BUFPAR[I]=_PAR[I];]
        _PAR[    0]= _OFSX; _PAR[    1]=  _OFSY; _PAR[2]= _OFSZ;
        _PAR[    3]=     0; _PAR[    4]=      0; _PAR[
```

▶７月号が「続・創刊10周年記念〜」だと思っていたら，８月号は「続々・創刊〜」ですか。いつまで続くんでしょう。ぞくぞくします。　中村　圭介(17)神奈川県

```
5]=     0;
        _PAR[_HEAD]=_SHEAD; _PAR[_PITCH]=_SPITCH; _PAR[_BANK]=_SBANK;
        FOR I=0 TO 3
        [
            _ZAHYO[I][0]=XV[I];
            _ZAHYO[I][1]=YV[I];
            _ZAHYO[I][2]=ZV[I];
        ]
        _LPCT[0]=4;
        @SETWD(&JO,3);
        @MAGIC(0);
        PX0=_DISP[0][0]; PX1=_DISP[1][0]; PX2=_DISP[2][0]; PX3=_DISP[3][0];
        PY0=_DISP[0][1]; PY1=_DISP[1][1]; PY2=_DISP[2][1]; PY3=_DISP[3][1];
        NZ=(PX1-PX0).*.(PY2-PY1)-(PY1-PY0).*.(PX2-PX1);
        IF ((_WSEL AND 1)==1) @MAGIC(1);
        IF ((_WSEL AND 2)==2) [ FOR I=0 TO 8 [_PAR[I]=_BUFPAR[I];] RETURN; ]
        IF (NZ.<.-4) [ FOR I=0 TO 8 [_PAR[I]=_BUFPAR[I];] RETURN; ]
        IF ((_WSEL AND 4)==4)
        [ @MAGIC(1); FOR I=0 TO 8 [_PAR[I]=_BUFPAR[I];] RETURN; ]
        NX=(YV[1]-YV[0]).*.(ZV[2]-ZV[1])-(ZV[1]-ZV[0]).*.(YV[2]-YV[1]);
        NY=(ZV[1]-ZV[0]).*.(XV[2]-XV[1])-(XV[1]-XV[0]).*.(ZV[2]-ZV[1]);
        NZ=(XV[1]-XV[0]).*.(YV[2]-YV[1])-(YV[1]-YV[0]).*.(XV[2]-XV[1]);

#ENDIF

#IF (_TILESW==1)  AND ((_GHIN AND 2)==2) AND (_FLOAT==1) AND (_THREE==1)

        @cvitf(plx,_LX); @cvitf(ply,_LY); @cvitf(plz,_LZ); @cvitf(lg,_LX);
        @mul(lg,lg,lg); @mul(ply,ply,ply); @mul(plz,plz,plz);
        @add(lg,lg,ply); @add(lg,lg,plz);
        @sqr(lg,lg);
        @cvitf(ply,_LY); @cvitf(plz,_LZ);
        @div(plx,plx,lg); @div(ply,ply,lg); @div(plz,plz,lg);
        @cvitf(ax,NX); @cvitf(ay,NY); @cvitf(az,NZ);
        @cvitf(lg,0); @cvitf(yg,0);
        @mul(ax,ax,ax); @mul(ay,ay,ay); @mul(az,az,az);
        @add(lg,lg,ax); @add(lg,lg,ay); @add(lg,lg,az);
        @sqr(lg,lg);
        @cvitf(ax,NX); @cvitf(ay,NY); @cvitf(az,NZ);
        @mul(ax,ax,plx); @mul(ay,ay,ply); @mul(az,az,plz);
        @add(yg,yg,ax); @add(yg,yg,ay); @add(yg,yg,az);
        @div(yg,yg,lg); @cvitf(lg,7);
        @mul(yg,yg,lg); LV=(@cvfti(yg)+7)./.2;
        IF (LV.<.0) LV=0;
        IF ((_GMASK AND 1)==1)
        [
            @MODE(2,0); @GRAD(LV*_SBLUE/7);
            @TRIANGLE(PX0,PY0,PX1,PY1,PX2,PY2);
            @TRIANGLE(PX2,PY2,PX3,PY3,PX0,PY0);
        ]
        IF ((_GMASK AND 2)==2)
        [
            @MODE(2,1); @GRAD(LV*_SRED/7);
            @TRIANGLE(PX0,PY0,PX1,PY1,PX2,PY2);
            @TRIANGLE(PX2,PY2,PX3,PY3,PX0,PY0);
        ]
        IF ((_GMASK AND 4)==4)
        [
            @MODE(2,2); @GRAD(LV*_SGREEN/7);
            @TRIANGLE(PX0,PY0,PX1,PY1,PX2,PY2);
            @TRIANGLE(PX2,PY2,PX3,PY3,PX0,PY0);
        ]

#ENDIF

#IF (_THREE==1) AND ((_GHIN AND 2)==2)

        IF ((_WSEL AND 8)==8) @MAGIC(1);
        FOR I=0 TO 8 [_PAR[I]=_BUFPAR[I];]
    ]

#ENDIF

#IF (_THREE==1) AND (_SINCOS==1) AND ((_GHIN AND 2)==2)

@KYU(MDX,MDY,MDZ,HNK)
    VAR I,J,K,L;
    ARRAY KYUX[3],KYUY[3],KYUZ[3];
    [
        I=0;
        WHILE(I<180)
        [
            J=0;
            K=@ISIN(I,HNK); L=@ISIN(I+_KSTEP,HNK);
            KYUY[0]=@ICOS(I,HNK)+MDY;          KYUY[1]=KYUY[0];
            KYUY[2]=@ICOS(I+_KSTEP,HNK)+MDY; KYUY[3]=KYUY[2];
            WHILE(J<360)
            [
                KYUX[0]=MDX+@ISIN(J,K);
                KYUX[1]=MDX+@ISIN(J+_KSTEP,K);
                KYUX[2]=MDX+@ISIN(J+_KSTEP,L);
                KYUX[3]=MDX+@ISIN(J,L);
                KYUZ[0]=MDZ+@ICOS(J,K);
                KYUZ[1]=MDZ+@ICOS(J+_KSTEP,K);
                KYUZ[2]=MDZ+@ICOS(J+_KSTEP,L);
                KYUZ[3]=MDZ+@ICOS(J,L);
                @CPOLY(KYUX,KYUY,KYUZ);
                J=J+_KSTEP;
            ]
            I=I+_KSTEP;
        ]
    ]

#ENDIF

#IF (_SINCOS==1)

@ISIN(KAKU,HAN)
  VAR   I,J;
  BEGIN
    WHILE(KAKU.<.0) KAKU=KAKU+360;
    KAKU=KAKU MOD 360;
    J=KAKU MOD 90;
    CASE KAKU OF [
        0 TO  89        I=_STABLE[J];
       90 TO 179        I=_STABLE[89-J];
      180 TO 269        I=-_STABLE[J];
      270 TO 359        I=-_STABLE[89-J];
    ]
    I=(I./.(32767./.HAN));
  END(I);

@ICOS(KAKU,HAN)
  BEGIN
    KAKU=@ISIN(KAKU+90,HAN);
  END(KAKU);

@IATN(XX,YY)
  VAR   X,Y,DATA;
  BEGIN
    X=ABS(XX); Y=ABS(YY);
    IF (X>Y) DATA=    _ATNTBL[(Y*64)/X];
    ELSE     DATA=90-_ATNTBL[(X*64)/Y];
    IF (YY.<.0)
    [
      IF (XX.<.0) RETURN(180+DATA); ELSE RETURN(360-DATA);
    ]
    ELSE
    [
      IF (XX.<.0) RETURN(180-DATA); ELSE RETURN(DATA);
    ]
  END;

#ENDIF

#IF (_THREE==1) AND (_SINCOS==1) AND ((_GHIN AND 2)==2)

@SCTL(I,J,POSY,SNP)
    ARRAY       PX[3],PY[3],PZ[3];
    VAR         K;
    [
        PX[3]=I-_KSTEP; PX[2]=PX[3]; PX[1]=I; PX[0]=PX[1];
        PZ[3]=J; PZ[2]=J-_KSTEP; PZ[1]=PZ[2]; PZ[0]=PZ[3];
        K=@ISIN(I,SNP);
        PY[0]=@ISIN(J,K)+POSY;
        PY[1]=@ISIN(J-_KSTEP,K)+POSY;
        K=@ISIN(I-_KSTEP,SNP);
        PY[3]=@ISIN(J,K)+POSY;
        PY[2]=@ISIN(J-_KSTEP,K)+POSY;
        @CPOLY(PX,PY,PZ);
    ]

@WAVE(XSIZE,ZSIZE,POSY,SNP)
    VAR I,J;
    [
        I=-XSIZE;
        WHILE(I.<._KSTEP)
        [
            J=ZSIZE;
            WHILE(J.>.0)
            [
                @SCTL(I,J,POSY,SNP);
                J=J-_KSTEP;
            ]
            I=I+_KSTEP;
        ]
        I=XSIZE;
        WHILE(I.>.-_KSTEP)
        [
            J=ZSIZE;
            WHILE(J.>.0)
            [
                @SCTL(I,J,POSY,SNP);
                J=J-_KSTEP;
            ]
            I=I-_KSTEP;
        ]
    ]

#ENDIF
```

▶ 7月号の裏表紙がワサビ味になった。　　山口　剛(18)東京都

▶ 全機種共通システムインデックス ◀

＊以下のアプリケーションは，基本システムであるS-OS“MACE”またはS-OS“SWORD”がないと動作しませんのでご注意ください。

1985

■85年６月号
序論　共通化の試み
第１部　S-OS“MACE”
第２部　Lisp-85インタプリタ
第３部　チェックサムプログラム
■85年７月号
第４部　マシン語プログラム開発入門
第５部　エディタアセンブラZEDA
第６部　デバッグツールZAID
■85年８月号
第７部　ゲーム開発パッケージBEMS
第８部　ソースジェネレータZING
■85年９月号
インタラプト　S-OS番外地
第９部　マシン語入力ツールMACINTO-S
第10部　Lisp-85入門(1)
■85年10月号
第11部　仮想マシンCAP-X85
連載　Lisp-85入門(2)
■85年11月号
連載　Lisp-85入門(3)
■85年12月号
第12部　Prolog-85発表

1986

■86年１月号
第13部　リロケータブルのお話
第14部　FM音源サウンドエディタ
■86年２月号
第15部　S-OS“SWORD”
第16部　Prolog-85入門(1)
■86年３月号
第17部　magiFORTH発表
連載　Prolog-85入門(2)
■86年４月号
第18部　思考ゲームJEWEL
第19部　LIFE GAME
連載　基礎からのmagiFORTH
連載　Prolog-85入門(3)
■86年５月号
第20部　スクリーンエディタE-MATE
連載　実戦演習magiFORTH
■86年６月号
第21部　Z80TRACER
第22部　magiFORTH TRACER
第23部　ディスクダンプ＆エディタ
第24部　“SWORD”2000 QD
連載　対話で学ぶmagiFORTH
特別付録　PC-8801版S-OS“SWORD”
■86年７月号
第25部　FM音源ミュージックシステム
付録　FM音源ボードの製作
連載　計算力アップのmagiFORTH
特別付録　SMC-777版S-OS“SWORD”
■86年８月号
第26部　対局五目並べ
第27部　MZ-2500版S-OS“SWORD”
■86年９月号
第28部　FuzzyBASIC発表
連載　明日に向かってmagiFORTH
■86年10月号
第29部　ちょっと便利な拡張プログラム
第30部　ディスクモニタDREAM
第31部　FuzzyBASIC料理法<1>
■86年11月号
第32部　パズルゲームHOTTAN
第33部　MAZE in MAZE
連載　FuzzyBASIC料理法<2>
■86年12月号
第34部　CASL & COMET
連載　FuzzyBASIC料理法<3>

1987

■87年１月号
第35部　マシン語入力ツールMACINTO-C
連載　FuzzyBASIC料理法<4>
■87年２月号
第36部　アドベンチャーゲームMARMALADE
第37部　テキアベ作成ツールCONTEX
■87年３月号
第38部　魔法使いはアニメがお好き
第39部　アニメーションツールMAGE
付録　“SWORD”再掲載とMAGICの標準化
■87年４月号
第40部　INVADER GAME
第41部　TANGERINE
■87年５月号
第42部　S-OS“SWORD”変身セット
第43部　MZ-700用“SWORD”をQD対応に
■87年６月号
インタラプト　コンパイラ物語
第44部　FuzzyBASICコンパイラ
第45部　エディタアセンブラZEDA-3
■87年７月号
第46部　STORY MASTER
■87年８月号
第47部　パズルゲーム碁石拾い
第48部　漢字出力パッケージJACKWRITE
特別付録　FM-7/77版S-OS“SWORD”
■87年９月号
第49部　リロケータブル逆アセンブラInside-R
特別付録　PC-8001/8801版S-OS“SWORD”
■87年10月号
第50部　tiny CORE WARS
第51部　FuzzyBASICコンパイラの拡張
第52部　X1turbo版S-OS“SWORD”
■87年11月号
序論　神話のなかのマイクロコンピュータ
付録　S-OSの仲間たち
第53部　もうひとつのFuzzyBASIC入門
第54部　ファイルアロケータ＆ローダ
インタラプト　S-OSこちら集中治療室
第55部　BACK GAMMON
■87年12月号
第56部　タートルグラフィックパッケージTURTLE
第57部　X1turbo版“SWORD”アフターケア
　　　　ラインプリントルーチン
特別付録　PASOPIA7版S-OS“SWORD”

1988

■88年１月号
第58部　FuzzyBASICコンパイラ・奥村版
付録　石上版コンパイラ拡張部の修正
■88年２月号
第59部　シューティングゲームELFES
■88年３月号
第60部　構造型コンパイラ言語SLANG
■88年４月号
第61部　デバッギングツールTRADE
第62部　シミュレーションウォーゲームWALRUS
■88年５月号
第63部　シューティングゲームELFESⅡ
第64部　地底最大の作戦
■88年６月号
第65部　構造化言語SLANG入門(1)
第66部　Lisp-85用NAMPAシミュレーション
■88年７月号
第67部　マルチウィンドウドライバMW-1
連載　構造化言語SLANG入門(2)
■88年８月号
第68部　マルチウィンドウエディタWINER
■88年９月号
第69部　超小型エディタTED-750
第70部　アフターケアWINERの拡張
■88年10月号
第71部　SLANG用ファイル入出力ライブラリ
第72部　シューティングゲームMANKAI
■88年11月号
第73部　シューティングゲームELFESⅣ
■88年12月号
第74部　ソースジェネレータSOURCERY

1989

■89年１月号
第75部　パズルゲームLAST ONE
第76部　ブロックゲームFLICK
■89年２月号
第77部　高速エディタアセンブラREDA
特別付録　X1版S-OS“SWORD”<再掲載>
■89年３月号
第78部　Z80用浮動小数点演算パッケージSOR OBAN
■89年４月号
第79部　SLANG用実数演算ライブラリ
■89年５月号
第80部　ソースジェネレータRING
■89年６月号
第81部　超小型コンパイラTTC
■89年７月号
第82部　TTC用パズルゲームTICBAN
■89年８月号
第83部　CP/M用ファイルコンバータ
■89年９月号
第84部　生物進化シミュレーションBUGS
■89年10月号
第85部　小型インタプリタ言語TTI
■89年11月号
第86部　TTI用パズルゲームPUSH BON!
■89年12月号
第87部　SLANG用リダイレクションライブラリDIO.LIB

1990

■90年１月号
第88部　SLANG用ゲームWORM KUN
特別付録　再掲載SLANGコンパイラ
■90年２月号
第89部　超小型コンパイラTTC++
■90年３月号
第90部　超多機能アセンブラOHM-Z80
■90年４月号
第91部　ファジィコンピュータシミュレーションI-MY
■90年５月号
第92部　インタプリタ言語STACK
■90年６月号
第93部　リロケータブルフォーマットの取り決め
第94部　STACK用ゲームSQUASH!
第95部　X68000対応S-OS“SWORD”
特別付録　PC-286対応S-OS“SWORD”
■90年７月号
第96部　リロケータブルアセンブラWZD
■90年８月号
第97部　リンカWLK
■90年９月号
第98部　BILLIARDS
■90年10月号
第99部　ライブラリアンWLB
■90年11月号
第100部　タブコード対応エディタEDC-T
■90年12月号
第101部　STACKコンパイラ

1991

■91年１月号
第102部　ブロックアクションゲームCOLUMNS
■91年２月号
第103部　ダイスゲームKISMET
■91年３月号
第104部　アクションゲームMUD BALLIN'
■91年４月号
第105部　SLANG用カードゲームDOBON
■91年５月号
第106部　実数型コンパイラ言語REAL
■91年６月号
第107部　Small-C処理系の移植
■91年７月号
第108部　REALソースリスト編
■91年８月号
第109部　Small-Cライブラリの移植
■91年９月号
第110部　SLANG用NEWファイル出力ライブラリ
■91年10月号
第111部　Small-C活用講座（初級編）
■91年11月号
第112部　Small-C活用講座（応用編）
第113部　MORTAL
■91年12月号
第114部　Small-C SLANGコンパチ関数

CREATIVE COMPUTER MUSIC

Creative Computer Music入門(11)
効率的な採譜のやり方

Taki Yasushi　瀧　康史

今回は皆さんからも要望が多かった採譜について解説していきます。なかなかひと筋縄ではいきませんが，これができると，楽譜の出ていない曲でも自分なりの音楽にすることができるようになります。

久しぶりのゲームミュージックね

こんばんはです。また風邪をひいてしまいました。おかげで例によって例のごとく，鼻の下が「豚のケツ」のごときにピンク色に肌荒れしてしまいました。うぅ〜ん。鼻かむと痛いよ〜。まあ，馬鹿は風邪ひかないっていうし，風邪ひいたんだから，私は馬鹿じゃないんでしょう（夏風邪は馬鹿しかひかないともいうが）。

そんなわけで，今回は鼻をかむと痛くなる日に聴く音楽を紹介したいと思います……なんてね。大嘘。

今回は久しぶりにゲームミュージックのCDをオススメしたいと思います。X68000ユーザーにはあんまり縁のないゲームなんだけどね（X1turboユーザーならあるよ）。なんとなくわかっちゃったかな。そうなの。なんといっても先日，かの「こしろん」さんともお会いできたことだし，「ソーサリアン」のシンフォニーアレンジ版をご紹介しちゃいましょう。

アレンジは羽田健太郎氏。CDそのものはずいぶん前から売っているから，目にした人も多いハズだよね。

このCDを聴いてどんなところが気に入ったか……。そうですね〜，この曲は厚みがあるんですよ。ライナーノーツには，リラックスして聴けって書いてあったけど，これは，気合を入れて聴くのがオススメ。どの楽器が，どの場所で，どのようにして使われているかをよく聴いておくと，たいへん勉強になります。

アレンジをするときに，ヴァイオリンってどこに使われているのかなーとか，ホルンってどこに使われるのかなーって思うことってあるでしょ？ 実際そういう勉強は，「曲を聴きながらやるのがいちばん」だと思うから。

そんなときに，これを聴きながらここでオーボエが，ここでホルンが，ここでヴィオラが……ってなるくらいに，それぞれの楽器の旋律やメロディの絡み合いをよく聴くとたいへん勉強になります。

そうそう，今回やる採譜の勉強にももちろんなりますよ。

では，今月分，始めましょうか。

採譜の心得

前回の予告どおり，今回は採譜（耳コピとか音取りともいう）の効率的なやり方についてお話ししましょう。

まず，いわなくてはならないことは，採譜とはそんなに難しいことではないということ。簡単にいってしまえば，採譜とはメロディをなぞり，リズムを取ることですから，歌を唄えて（メロディをなぞる），手拍子を叩いたり，曲のノリがわかれば（リズムを取る），採譜という仕事は造作もないことです。

いつも思うのですが，採譜が難しいと思い込んでいる人は，たいてい「完璧なコピー楽譜」を作ろうとしています。できた楽譜は売るわけでもないし（著作権上売ることはできませんし），バンドなどの個人の楽しみに使ったり，シーケンサに入力して友達に聴かせて楽しんだり，多く見積もっても投稿するぐらいですから，そんなに力む必要はありません。

また，よくこのような思い込みを持つ人がいるんですが，必ずしも，

　　完璧にコピーされた楽譜
　　　　　＞適当にコピーされた楽譜

とはならないことを，肝に銘じておいてくださいね。完全にコピーされた楽譜よりも，自分色を出した楽譜。そう，コピーをするときは，まったく同じではなくて，多少自分でアレンジを加えるつもりで採譜してみてください。

それはもちろん，完全にコピーする練習をしたほうが耳は鍛えられます。聴こえ難い音まで聴き取ろうとするわけですから当然です（私もときどきバンド仲間に頼まれて，完コピすることがあるのですが）。

でも，逆に自分なりにこんな感じかな？って適当に仕上げていったほうが，たとえ最初はうまくできなくても，慣れてくると，早い人は4，5曲コピーするだけで自分の色でアレンジを加えた自分なりのコピーが採譜の段階で作ることができます。さらに慣れてくると，耳コピしながらオブリガードを思いついたり，スピーディなアレンジが展開できます。

要するに，このような方法で採譜をするということは，すなわち，アレンジをするテクニックを身につける近道でもあるんです。

どうですか？ まあ，そんなに難しいことじゃないですから，気楽に考えて進めてみましょう。

採譜の3カ条!?

とりあえず，例を探してきましょう。いきなり難しい曲を選ぶと最初からつまずいてしまうので，最初は自分で聴いてみて「耳コピできそうな曲」を探してみます。耳コピしやすい曲というのは，だいたい次の3つに該当しているものです。

1．メロディがハッキリしている
2．リズムがハッキリわかる
3．ベースラインがハッキリわかる

とまあ，曲の支柱になる3つが簡単にわかれば，その全体像がつかみやすいということになるわけです。あと，このほかにも強いていえば，

4．コードが取りやすい

という項目が入ります。ただ，1〜3の項目は，耳で聴き取る能力がほとんどを占めるのに対して，4は理屈で割り出すことができます。それに，1と3がわかってしまえば理論的にコードは割り出せるので，4はこの際除外して考えていきましょう。

このほかの曲の構成音は，適当でかまいません。聴き取り難い音は「みんなどうせ聴こえてない！」と，いい加減ですが割り切って考えてください。それがどうしても

気になりかけてきたら，耳が肥えてきた証です。気になる音はどんどん付け加えていきましょう。

なんとなく話がそれてしまいましたが，これら3つ（4つ）に該当する曲を選んでみます。考えてみると，邦楽のポップスなどは，かなり聴き取りやすいラインにあります。最近のものでいえば（流行を追う人からみれば最近ではないのかもしれませんが），槇原敬之の「どんなときも。」などは，簡単に耳コピできそうな曲ですし，またアレンジしやすい曲でもあります。

それから，別の意味でゲームミュージック，特に古めの曲などは，同時に鳴っている音が決まっているので，わかりやすいといえばわかりやすいです。ただ，妙に速い分散和音（それもゲームミュージックではとくに高音に多い）があると，耳で追えなくなってしまって，逆に挫折感を味わうことになってしまいます。

原曲を聴いたことのない人には申し訳ないのですが，私もちょっと前にズームのファランクスのオープニングの採譜を友人に頼まれてやってたんです。ところが，最初のイントロのオケヒのあとの，方形波のような分散和音がうまく聴き取れなくて，頭を抱えてしまいました。のっけからつまずいてしまったんですね。ほら，あのピロロピロロピロロピロロっていうやつですよ。悩んだ末，裏ワザに走ってテープデッキの1/2再生機能を利用して(テープスピードが1/2になる＝周波数が半分になる＝オクターブが1下がる！）取ってしまいました。

とまあ，私事に走ってしまいましたが，要するに最初は難しい曲はやらないほうがいいってことです（強引な攻め方だな)。

採譜については今回だけで話し終えるつもりは毛頭ないので（この続きは来月かもしれないし，再来月かもしれないし)，今回はお手軽に適当なゲームの曲から拾ってみましょう。

最初，コナミのグラディウスなんかが簡単そうでよいかなって思ったんですけど，採譜してみたら，全部同じ音色なので意外と大変だったんです。

そこで，ナムコのメルヘンメイズから，「ビックリの国」の曲をサンプルにしようと思って，楽譜まで書いたのですが，版権が下りなくて，結局，コナミのアーケードシューティング「ゼクセクス」のステージ1の曲を選んでしまいました。

曲の選考理由として，この曲はリズムが全編通してだいたい一定していてわかりやすいことと，メロディがさほど難しくないこと，さらにベースラインも同じ音が多く取りやすい，などが挙げられます。テンポはやや速めですが，前のほうで述べた採譜の3カ条にピッタリ当てはまるでしょう？

あ，原曲を知らない人はごめんね。どうしても，みんながみんな知ってるっていう曲はないものだから……おハガキ見ても，ゲームミュージックにしろという人もいるし，そうじゃないほうがいいっていう人もいるし……。

拍取り

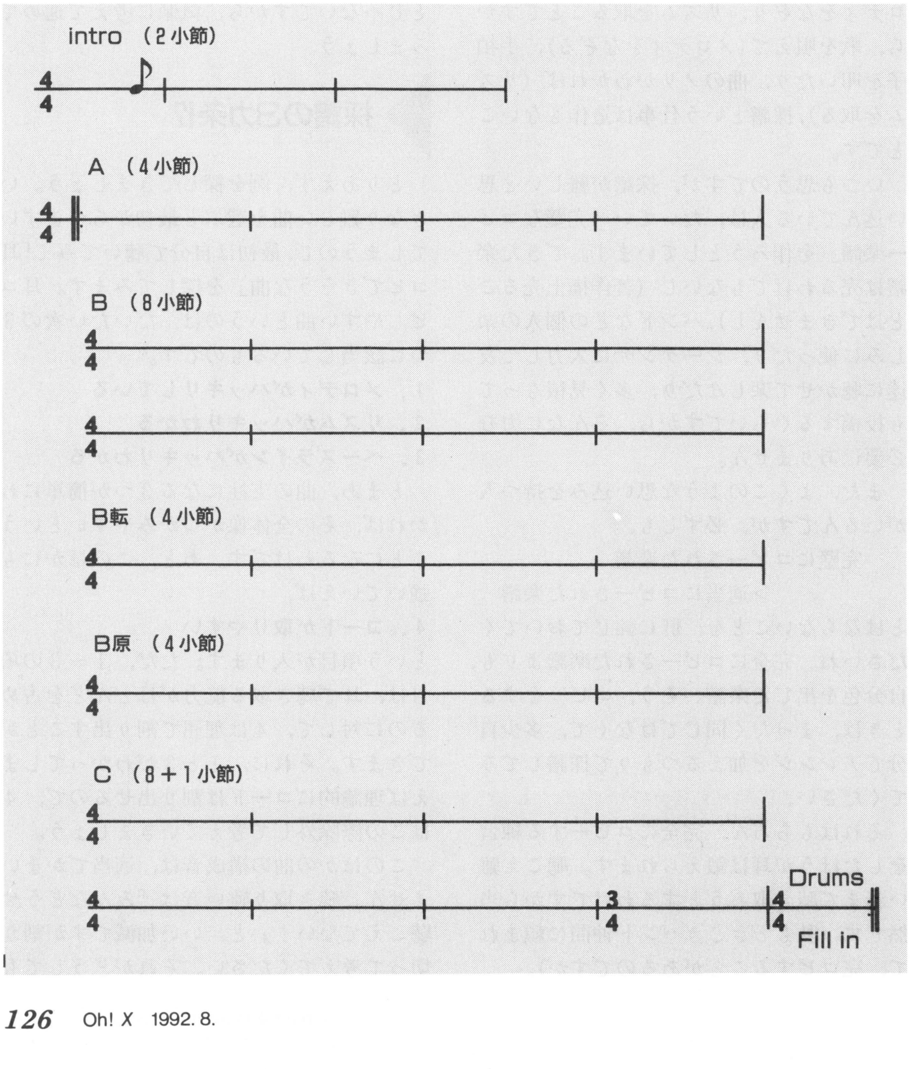
拍取りといっても，たいしたことはないのです。手拍子が取れればマル。そのうえ，この曲が4分の4，すなわち4拍子だとわかれば二重マルです。.

さっきは拍子を数えるために聴いてみました。最後から2番目の3/4拍子のところ以外は一貫して4拍子なので，そんなに問題はないと思われます。まあ，細かいところまでこだわって聴いてみると，ところどころ裏打ちをしていたり，イントロのアタマの部分が前の小節に半拍だけくっていたりするんですけど。でも最初にいったように，今回は「完璧なコピー楽譜」を作ることを目指してはいないのです。ですから，こういった聴き取りづらい「飾り」的なところは，聴き取れなくてぜんぜんかまいません。この曲はそういった細かいことを抜きにして，採譜の3カ条にほぼ沿っているということでサンプルにしたのですから。

普通，こういった曲はバックのドラムなどをよく聴くとわかります。このような曲をいわゆる「アウフタクト」というのですが，アウフタクトは最初からドラムを裏打ちしないハズなので，最初の強拍の音（バスドラム）を聴いていれば，だいたいわかります。たいてい1拍子目なんですが，聴いてみればわかるでしょう。

ここで図1を見てください。これはこの曲の小節が何小節あって，どのように区分されているかがわかるように取ったメモです。実は，この曲のように短い曲なら一気に楽譜に小節線を書いてもかまわないのですが，長い曲ではそうはいきません。小節の数と曲の構成（この曲では構成はたいしたことないですけど）をしっかり把握する

図1



ために，最初にこのような図を描く癖をつけましょう。そうそう，ここでいう曲の構成とは，この曲では，Intro，A，B，B転調，B原調復帰，Cと分類できます。それぞれ，図1もしくは楽譜を参考にしてください。

この曲にかぎらず，たいていの曲は4の倍数（もしくは2の倍数）で曲の節目が区切られています。もっとも，この曲は最後に1余りがつきますけどね。おそらく，ループする曲なので，単調になることを避けるためにやった行為なのではないかと思いますが。それでも，たいていの曲はやっぱり4の倍数なので，このことを参考にしながら拍取りを行ってください。

メロディ&オブリガードを取る

はてさて，拍が取れたら次はメロディです。別に最初にリズムを拾ってもいいんですけど，ほとんどの人はいちばんメロディが追いやすいと思うので，ここではメロディを取ることにしましょう（複雑で耳では追いにくいメロディの場合は，ベースから取って，スケール，コードを求めて，理論的&耳で取るのがわかりやすいです）。

メロディは，たいていいちばん高い音で綴られます。都合のいいことに，もっとも人間が聴き取りやすい音は高い音ですから，近くにある鍵盤などを奪ってきて，弾きながらドンドン聴き取って採譜してしまいましょう。

鍵盤があっても聴き取れない人は，多少薄情だとは思いますが，ほんとに鍛えるしかありません。CDなどをガンガン聴いて耳を鍛えてください。……とまあ，これではあまりにも薄情ですから，ソルフェージュ練習プログラムなどをあとに作っておく(作ってもらうという話も)ことにしましょう。

それでサンプルなんですが，デチューンがかかっていて，一聴ではなかなかわかりません。救いといえば，オブリガードがないに等しいので，その点は惑わされなくて楽だということでしょう。

デチューンのかかった音は，最近流行のディストーション系のギターのメロディを取るための勉強になりますから，無駄にはならないでしょう。

最近はいろいろ便利になったもので，グライコつきのステレオなんかがあると，いま聴いている楽器（パート）にいちばん近い周波数に合わせて，聴き取りやすくすることができます。また，CDなんかはいくら聴いても擦り切れませんし，ものによっては部分連続再生ができるので，かなりの手助けになるでしょう。

もっとも，聴いてみてどうしてもわからないところは，いまはおいといて，コードを乗せたときに取っても結構です。

リズム

曲によってリズムを担当している楽器が違います。たとえば，先月のサンプルでは，リズムはコントラバスのピチカートで刻んでいました。ポップスやロックなどの最近の音楽では，ドラムスがその大半を占めています。

よく，アマチュアバンドの人で，耳コピするときに寸分の違いもなくドラムをコピーしようとする人が多いですけど，なにもそこまで力まなくてもいいでしょう。なぁに，プロでもなし，自分のオリジナルにしてしまえばいいんです。というか，プロの人のほうが，どちらかというと自分の味やオリジナル性を出したがるんですけどね。まあ，気楽にドラムなんかはフィルインだけ押さえておけば，普通の人は気がつかないやっ，て考えていればいいんですよ。1992年3月号の私の記事でも見ながら考えてください。

この曲のリズム担当は基本的にはドラムス，パーカッションとしてのオケヒでしょう。どちらもあまりすんなりとしたものではありませんが，それほど難しいものではないでしょう。

ベースノート

ベースノートを一瞬で取ることが難しくても，ベースを担当している楽器の音は，ドラムなどのパーカッションを抜いた音の中で耳に聴くいちばん低い音ですから，簡単に取れると思います。それに，最近のオーディオシステムではバスブーストの機能がついているものが多いですから，それらのツマミをきゅっとひねるだけでよく聴こえる音になりますしね。

さてベースノートが，どんなものかといいますと，簡単にいってしまうとそれはコード中でいちばん下の構成音です。当然ですが，ベースが演奏する旋律とはおおいに関係があります。ベース音というのは，曲の中でも直接体に振動して（低い音ほど耳に聴こえるよりも体に感じる）くるので，誰でもすぐ感じることができるでしょう。

したがって，そのベースの関係するベースノートが曲中で意味する役割は，文字どおりベース（基盤）なので，たいへん重要です。これには，曲の雰囲気の決定，発展の決定，すなわちコード進行の決定ということになります。

そこで，その重要なベースノートを，実際のベースの旋律から割り出す方法を考えてみましょう。

たとえば，ベース(を担当している楽器)が同じ音を1/2小節から1小節程度の長さで繰り返して演奏している場合，それはもうたいていそのままベースノートです。それが，全音符だろうと，8ビートの8分音符で刻もうと間違いなくベースノートです。それから，このベースノートが切り替わるときは，まず間違いなくコードが変わるときです（いまここでは関係ないですけど，念のため)。まあ，これらのベースの旋律によって構成されているベースノートは特に問題ないでしょう。

問題は動くベースです。ポップス系の曲では，動いたといってもせいぜいコード構成音を経過音や刺繍音などの比和声音で結んであるだけですから，たいしたことはないのですが，クラシック系ではベースもオブリガードの旋律を演奏してるので，ベースノートは一発ではわからないのです（慣れの勝負ですけど)。もっとも最初からクラシックの楽譜を取ろうとするほうが間違ってるといえば間違ってるんですけど。いずれにしても，簡単な曲を何度かやってるうちにわかるようになるでしょう。

単純なベース音が取れないという人は，おそらくいません。取れているんだけど，だんだん見失うという場合がほとんどです。いろんな曲のベース（いちばん低い音）を耳で追っていけば，それほど苦労なしに取れてしまうんではないでしょうか。

動くベースラインからベースノートを取るには，ある程度の法則性を見つけるのがいちばん簡単，かつ妥当案です。ポップス系の音楽では，たいてい複雑な動くベースでも，ある程度の一定のベースのパターン(リフ)を繰り返す場合がほとんどです。そのなかから強拍にある音，特に最強拍（小節の1拍目）などの音は，たいていベースノートといってもいいと思います。これを箇条書きにすると，

1．小節最初の拍であるか

2．小節中いちばん多く出てくる音か

となります。もし，1，2に該当する音がベースラインから現れたら間違いなくそれは，ベースノートだと見て結構です。1と2とそれぞれ満たすものが違う場合，要す

るに最初の拍の音とたくさん出てくる音が違う場合は，1の小節中の最初の拍に注目してください。

サンプルの曲ですが，これがなかなかズルい（？）ことをやってのけてます。はっきりいって，最初はこんなベースラインは取れなくてかまいません。ぜーんぜんかまいませんから気にしないでください。

調性（スケール）を探る

いま流れている曲がどんなスケールで流れているか，すなわちトニックはなんというコードかを探すのは，これは聴いてるだけでは，なかなかピンときません。

そこで，まず絶対音感が身についている人は，曲の節目のトニックを探してください。トニックというのは曲の節目には必ずやってきますので，それを追っていけばスケールを取ることができます。

それでは絶対音感を持ってない人はどうしましょう？　ソルフェージュプログラムでも作って練習しますか？　ってこれではあまりにも薄情。だいたい，演奏する楽器（ギターとか）によっては絶対音階なんて全然身につきもしないでしょう。スケールをCもしくはAmということにして，無理やり取ってしまいますか？　それはあんまりよい案とはいえませんよね。

そこで，さっきやったベースノートを利用することによって，スケールを導き出す方法を考えてみましょう。このベースノートの音ですが，コードの構成音のうちなんの音だかわかりますか？　前回か前々回，ベースノートはコードの根音（ルート音）か，あっても5度の音だといいましたよね。ということは，最初の強拍のベースノートがトニックのルート音（もしくは5度の音。でも2つにひとつならなんとか選べますよね）になる可能性がかなり大きいことになります。もし，ここで最初にトニックがこない曲があったとしても，曲の節目では必ずトニックで終わって（るに違いないハズ）ますから，探すことは意外に容易だということになります。

もちろん，曲のスケール＝トニックのコードですから，トニックのコードがわかれば，曲のスケールがわかったことになります。

また，ベースノートだけしかわからずコードがわからない場合，最初のベースノートがAなら（これだけの条件で）考えられるのはAmかA(maj)です。スケールの根音さえわかれば，あとはフラットしている音，シャープしている音を探していくだけで調性は簡単にわかりますよね。いまの例なら，CもしくはFにシャープがかかってるとメジャー（長調）ですし，かかってないとマイナー（短調）です。こうなるともうスケールはわかったも同然でしょう。

（注意：借用和音には注意する必要がありますが，最初から借用ばかり使う意地悪な曲は……あ，「バナナ」が……あるかもしれません。う～む）

コードを取る

採譜には，コード取りが必要です。なぜなら，それは簡単。だって，音取りが楽になるから。コード上に乗った分散和音（このことについてはあとで詳しく説明しましょう）などは，コードさえわかっていればまず間違いなく取れます。それに，コードさえわかっていれば，耳で聞こえないほど隠れた音でも，適当においてはめることができます。

さてと。ベースノートがすでにわかっていますから，あとは簡単です。

何も知らない人は，コードを取れというといきなりコードを「耳で聴き取ろう」とする人がいます。え？　自分がそうだった？　知らなくてもOK！　いまから知ってしまえばなんの恥もありません。そうそう，コードを耳で取るなんて，ソルフェージュでそうとう耳を鍛えるか，普段コードを意識しながら曲を聴く癖をつけないかぎり，なかなか難しいことです。まあ，確かに一部の人たちは，曲を聴きながらコードがドンドンわかってしまうのですが，これはそんな人たちのための講座ではなく，むしろそういったことがわからない人たちのための講座なので，ここではわかってしまう人たちは無視して，理屈でコードを求めていきましょう。

さっきもいったとおり，ベースノートというのは，基盤の音ですから，コードでは必ず乗る音になります（若干の嘘あり）。基盤の音は，たいていコードのルート音になっている可能性が大なんです。あ，ルート音というのは，コードを構成する基本の音，すなわち，第1度の音のことです。

たとえば，CmならC，Amaj7ならA，DmならD，G7ならG。そう，Cdim7などの，Cの音です。

もちろん，この音はたいていいちばんコードに乗りやすい音（というかそのもの）ですから，外れる音ではありません。

（注意：実際には，根音省略形というヤなヤツもいます。が，ここでは無視無視。そんなの，ポップス系の曲じゃめったに出てこないよ）

ええ～と，なんだっけかな。そうそう。それから，スケールはわかってるはずですから，ダイアトニックトライアドコードを導いておきましょう。え？　ダイアトニックトライアドコードってなんだかわからない？　う～ん。確か連載1回目か2回目あたりにやったからなあ。忘れても無理はないか。

ダイアトニックトライアドコードというのは，スケールの構成音，たとえばC(maj)というスケールならCDEFGAB。DmならDEFGAB♭C）をそれぞれ根音とする3和音です。すなわち，スケールC(maj)なら，

C Dm Em F G Am Bdim

がダイアトニックトライアドコードということになります。詳しい説明は1991年10月号から12月号あたりを見てくださいな。

で，このダイアトニックコードの中で最頻繁に出てくるコードが，トニック，ドミナント，サブドミナントでしたよね。C(maj)というスケールなら，

C　F　G

の3つのコードのことを指します。

これらがなぜ大事なのかは，この3つのコードの構成音だけで，スケール全部の音を網羅するとか，いろいろ理屈は以前述べましたよね？　当然ですが，その知識はここでも生きてきます。

サンプルの曲はC(maj)というスケールですから，C，F，G（どれもメジャーコードね）が頻繁に出てくることがまず予想されます。GはよくG7として使われることも前にいいましたね。

この曲のベースラインは途中で暴れたりしてますが，ベースノートはブロックA，B，Bの原調復帰部（以下B原と略）では一貫してCです。ブロックBの転調部（以下B転と略）でスケールがBに落ち，ベースノートがBになります。そして，曲の展開部（サビ）のCで，それまでCで保ってきたベースノートがB♭，A，A♭，Gと動き出し，そしてまたAに戻りループする構造になっています。

実のところ，この曲はいろんなところで偶成和音が使われています。だいたい，ブロックAではベースノートはCですが，コードはG7。これは，イントロでトニック―サブドミナントときているので，ドミナント7thからAメロでは開始しているのですが，これを解説しても，まだしっかり偶成和音を教えてない現在では，余計にわけが

わからなくなるだけなので，今回は比較的構成がわかりやすいブロックCだけ説明しようと思います（建て前はそうだが，本音はいまから偶成和音についてやる時間がないだけのこと……ごめんね，って今回謝ってばっかり……）。

当然ですが，ブロックCのコードを弾くところ（132ページの楽譜中の1段目から）はまだ空白だという状態での話ですよ。埋まってたらそれをそのままコードにすればよいでしょうに。

まず，ベースノートはB♭，いきなり変なやつが出てきます。でも焦らずに。前（B原の最後）をちょっと覗くと，ベースラインは暴れていますが基本的にはベースノートはCできています。それから，Cの1小節目のメロディはE～EFGと奏でられています。当然ですが，コード構成音中にEが入っているのは妥当な考え方でしょう。

EとB♭の両方が含まれているコード。そんなの，ほとんどありません。あってもややこしいもの（C#dimとか）だけです。そこで，前のベースCを見ます。そして次の小節のAを見ます。するとこれは，経過によるベースなんじゃないか？　という考えが浮かんできます。講座中にこんなのを持ってくるのはほとんど詐欺に近いのですが，B♭を含むコードで根音がスケール上に乗っているものは，実はトライアドコードではないのです。そこで，クォードにしてみます。

さて，ダイアトニッククォードコードなどはまだ例にも出してないのですが，ここで一気に簡単に説明してしまいましょう。

まず，ダイアトニッククォードコードとはどうやって作るかといいますと，これは7thコードにするのが一般的です。これらの基本的説明は1991年12月号を，詳しい説明はまた今度ということにして，とにかくスケール上に7thコードを置いてみましょう。すると，

C7 Dm7 Em7 F7 G7 Am7 Bdim7

となるわけです。これらの図は今回は省略してしまいますが，この中でB♭が入るのははC7だけです（トニック代理）。

まあ，そんな理由で（わけがわからない人にはこじつけに見えるでしょうが，これはのちほど，偶成和音を説明するときに詳しくお話しします）ここはC7なのです。

次の小節はベースノートがA，メロディがEですから，これら2つが構成音として含まれているダイアトニックトライアドコードは，Amしかありません（トニック代理）。

その次の小節は，ベースノートはA♭，また1拍目（強拍）の音がE♭そして，3，4拍目の長くのばす音はA♭ですから，これはどう考えてみても，コードはA♭(maj)です。ただし，A♭(maj)はスケール上のコードではありません（これでもコード取りとしてはよいのですが）。そこで，さっきと同じようにダイアトニッククォードコードの7thを取ってみますと，F7がA♭(maj)のすべての音を包含していることに気がつきます。よって，これはF7です（サブドミナント）。

ここでは実に簡単にコードが割り出せてしまいましたが，このようにどっちか迷ったら，実際に弾いてみて確かめるのがいちばんですし，ほんとはこんなところまで理屈で求めなくても，実際は迷ったら聴くことがいちばん。なぜって，音楽をやってるんですもの……。

このままのペースでは全部説明つけなくてはならないので，このくらいでやめておきましょう。

まとめ

まとめあげると，採譜という作業は今回やった拍取り，リズム取り，ベースノート取り，メロディ取り，コード取りといった作業に分類されます。今回やった順序は私が勝手に考えた順序で，別にこうでなくて

今月のうんちく

7月号のLIVE inに載ったMZ-2500版，エリック・サティ（E.Satie）の「ヴェクサシオン（Vecsation）」について，面白いうんちくがあったのでひと言。

ヴェクサシオンは世界で最も長いといわれている曲で，なんと全曲演奏時間は18時間ぐらいになるという長作です。内容は1分たらずのメロディをえんえんと840回も繰り返すという，とんでもない曲です。

実はこの曲，作られただけで演奏会はなかった……なんてことはありません。1893年に作曲されたこの曲は，1963年，ジョン・ケージ（ピアニスト）を中心とする数人のピアニストのチームにより，ブロードウェイでちゃんと演奏会が行われました。最初で最後の演奏会だったんじゃないかな，きっと。

もちろん，1ループ1分で終わってしまうような曲を，えんえんと18時間も聴衆が聴いていられるはずはありません。そのときの新聞記事によると，聴衆はやっぱり好き勝手なことをしていて，寝ていることはもちろん，食事をしていたり，演奏中の出入りもこと激しかったそうです。そうそう，演奏会で演奏中に席を立ったり，会場を出たり入ったりすることは，本当は大変失礼なことなんですよ。

ピアニストもピアニストで，勝手気ままに個人的解釈を付け加え，即興でアレンジをしながら演奏したそうです。なるほど，同じ節を同じように演奏してたんじゃ，ピアニストもたまったもんじゃないですからねぇ。

さて，気になる入場料は5ドル。となると，その当時，円はサークルというわけで，1ドル360円だったはずでしたから，だいたい日本円に直して1,800円。当時の日本円の価値といまの価値を比較すると，う～ん，5,000円ぐらいかしら。そうなると，高いんだか安いんだか。はたまた，損なような得なような気がしますね。

ところが面白いことに，演奏会では20分我慢して聴くごとに，お金が5セントずつ返金されたそうです。全部聴き通すと，20セントのごほうびがもらえたそうですから，頑張って聴いた人もなかにはいるかもしれませんね（根性のある人だ……）。

それから，CDはないと書いてあったけどそれは嘘(確かにCDはないかもしれないけど)。実はその昔，レコードは発売されていたみたい。いまはどうだかわからないけど，ラインペルト・デ・レーウ（ピアニスト）による「全曲録音盤」なるものが存在したらしいから，ひょっとしたらどこぞの骨董品屋に眠ってるかもしれません（手に入れた人は編集部にご一報ください。いるとは思えんが……）。もちろん，全部ベタでそのまま入ってたら，とんでもない枚数になっちゃうから，圧縮して……じゃない，なかば卑怯だけど，35回分の演奏（この35回分の演奏という表記は誤りだと思うんだけど）が入っていて，24回連続して聴くのが筋だったようです（それがほんとに筋かは知らんけど……，あ，35×24＝840ね）。

私がなるほどな～って感じたのは，作曲者サティの意図。ただ単にギャグで（それも聴衆を驚かすためだけに）作られたわけではなく，ちゃんと意味があったんだからすごいと思う。

当時の音楽の聴き方は，作品全体を踏まえて聴くことが重要視されていたため，作曲者は作品全体の構成の完成度，芸術性を求めて，いわばそれを多分に意識しながら作っていました。

そこでサティが考え出したものが「表現・解釈を放棄した音楽」。短いモチーフの繰り返しを使うという技法は，いわば「表現性の消去のために用いられた」というわけです。その思想を具体化したものが「ヴェクサシオン」だったんですね。誰ですか？　これはSIONの曲だっていっていた人はっ！

蛇足ながら，ジョン・ケージは，知る人ぞ知る「4分33秒」（ピアニストがピアノの前に座り，4分33秒の間じっとしているだけというもの……なんなんでしょうな）の作者です。

もっとも，あのヴェクサシオンも調べてみれば意図があったのですから，この4分33秒も意図があるかもしれませんね。

それでは，今回はこんなところで幕を閉めさせてもらいましょう。

ではまた。

はならないといったものではありません。

今回，このような内容の講座を書くことになったので，バンド仲間や編集部でも善ちゃんや浦川さんなど，いろいろな人に採譜するときはどうする？　って聞いてみました。やっぱり人それぞれ，さらに曲によりけり。リズムを最初に取ったほうがわかりやすい曲もあれば，メロディにインパクトがあるからメロディから取ったほうがわかりやすい曲もある。いつもメロディを追って聴いてるからメロディを最初に聴き取る人もいれば，リズムから必ず取るという人もいる。それに，拍取りなんてしないよーって人もいる。

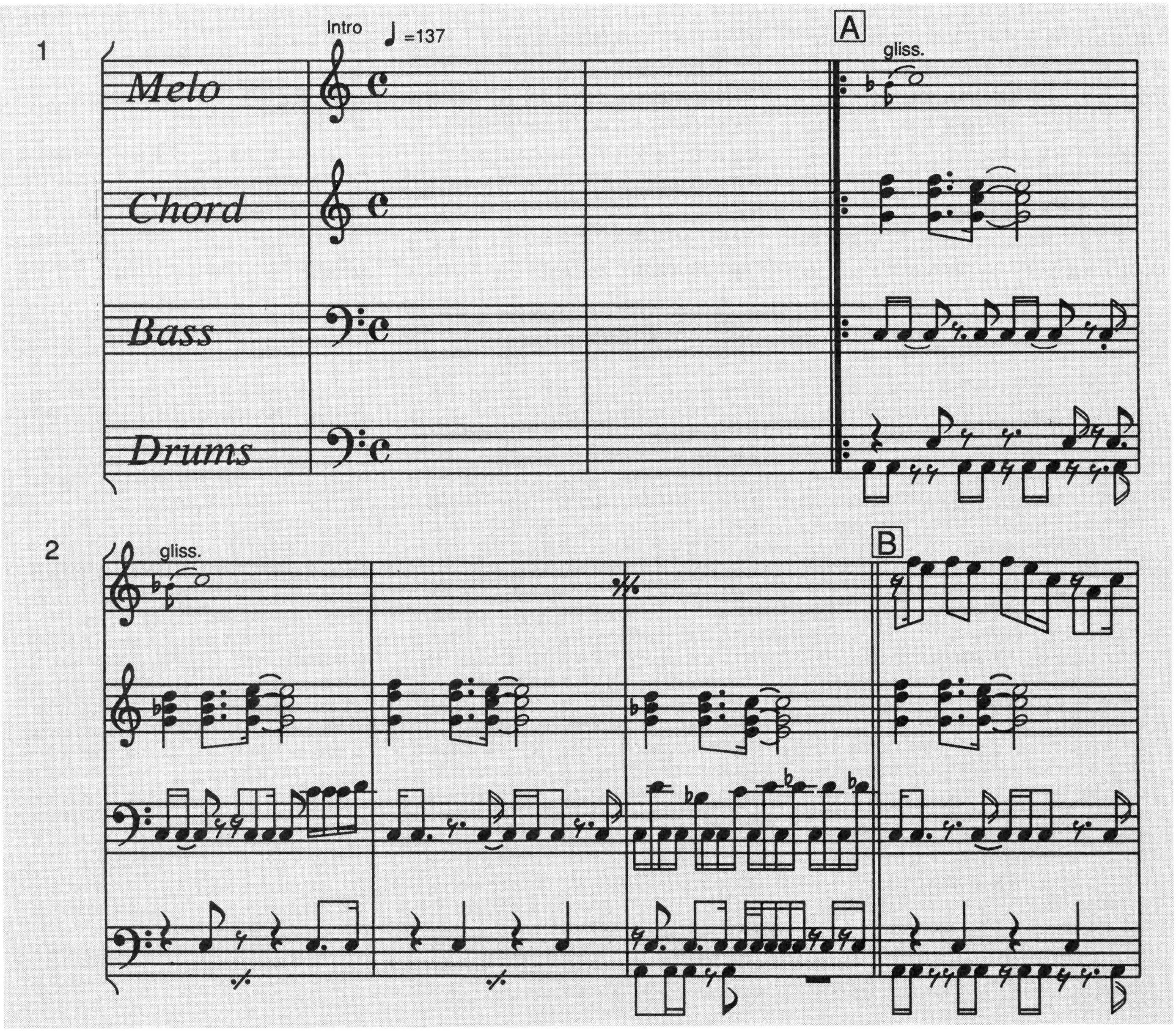
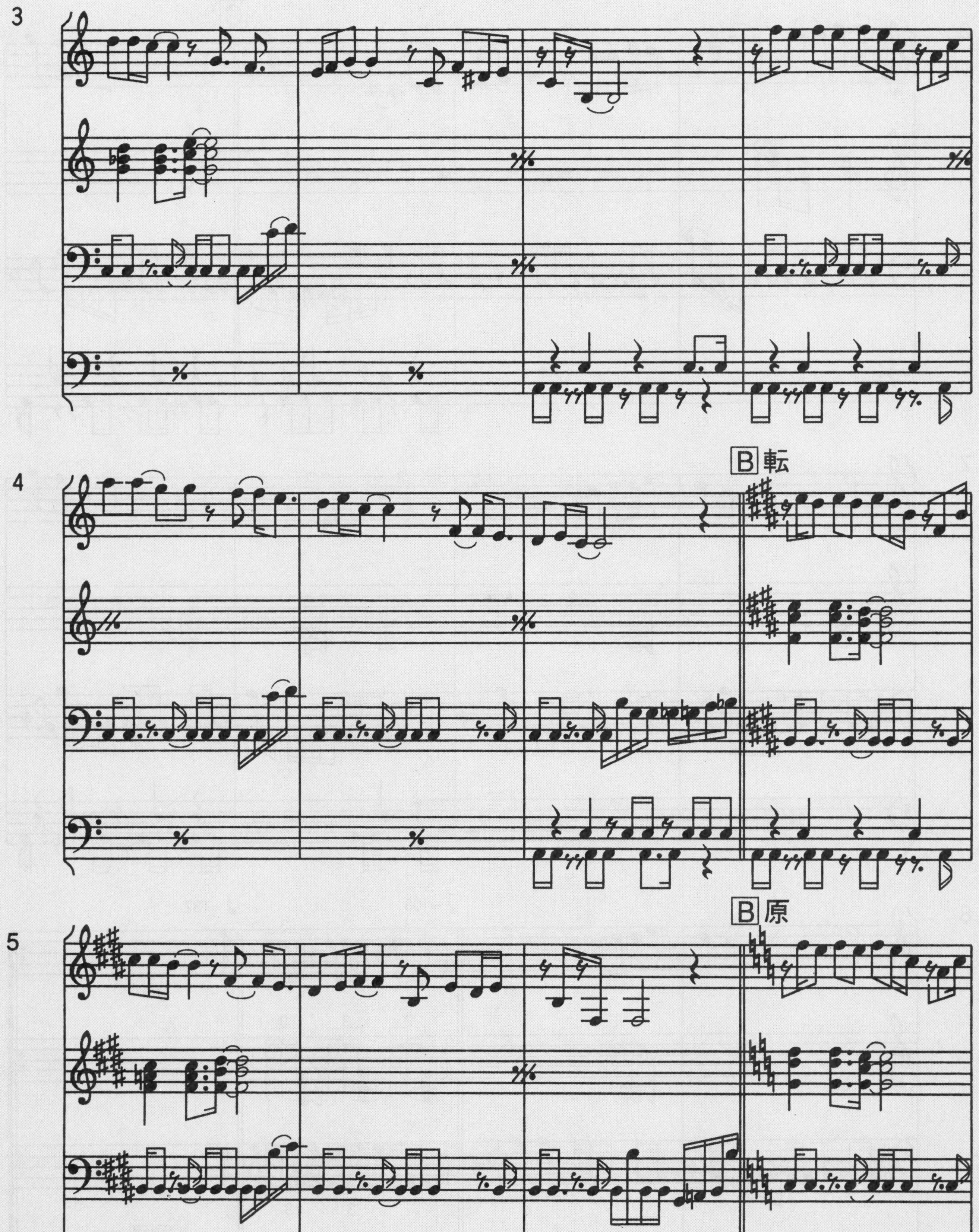
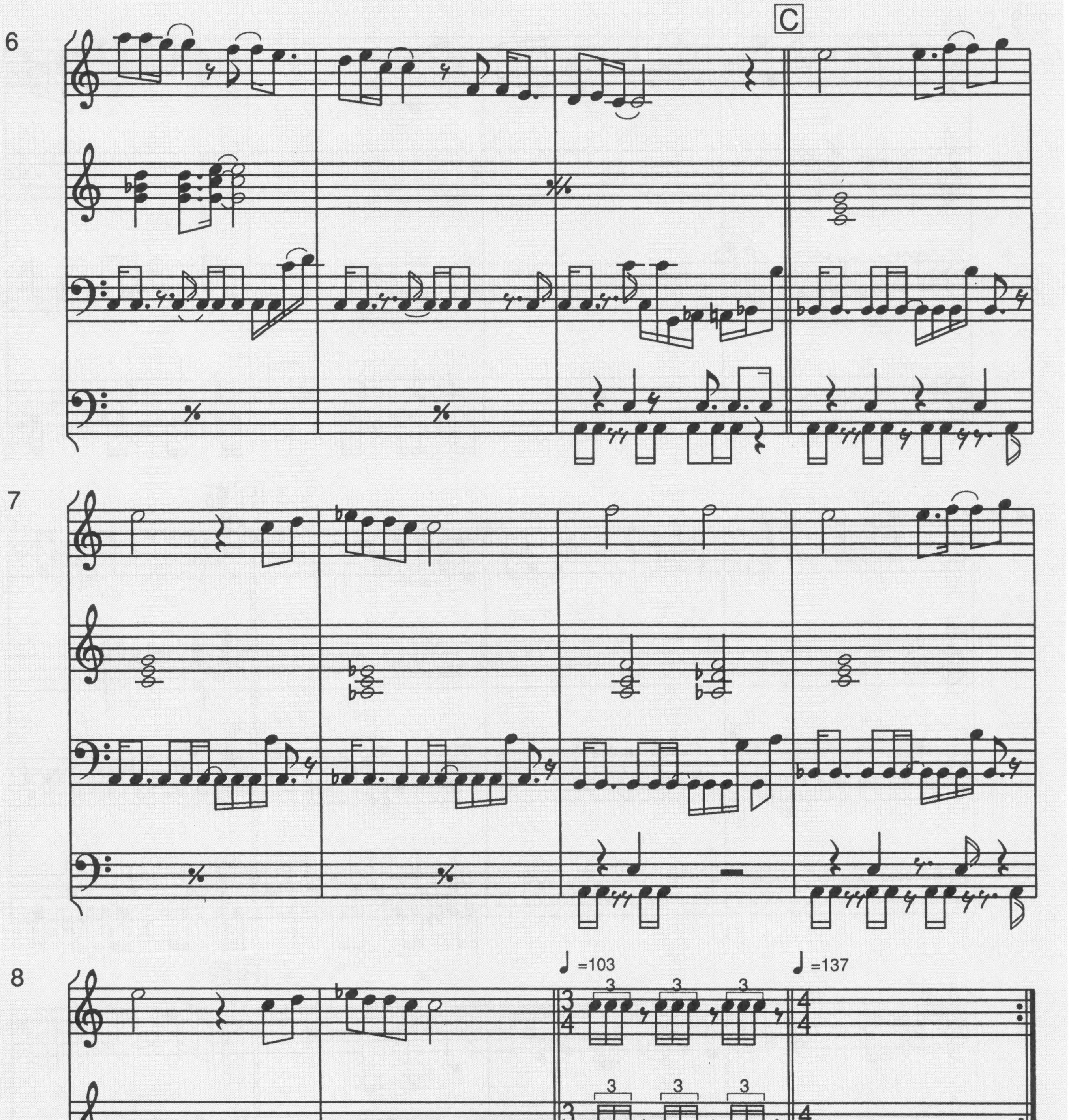
傾向を見ると，ドラマーはドラムから取るのが多く，ギタリストはギターから取るのが多いようです。やっぱり自分のやっている楽器がいちばん取りやすいってことですかね……。

はてさて，いい忘れてました。当然ですが，この楽譜はどっかから持ってきたものだったり，コナミさんから提供されたものではありません。だから，いろんなところで取り逃がしてる音があるかもしれません。それとイントロの部分や，ところどころカザリなどが抜けています。まあ，曲の大筋をつかむことを目的として採譜をしたので，そこのところはご容赦ください。ということで，そのあたりは皆さん自身でアレンジするなりしてみてください。できたら，投稿なんかもしてくれるとうれしいです。

そうそう，メロディの件ですが，メロディも理屈で取ってしまえば，2月号でやった，非和声音を駆使することによって，一応ある程度は求められます。でも，でも，でもさ。メロディぐらいは，自分の耳で聞き取ろうよ。なんだかんだいってもさ，採譜のカギは自分の耳。成人してしまっても，かなりのレベルまで慣れで取れるようになるはずです。それに，10代の人なら，いまのうちに耳を鍛えておいても損はないと思うよ。

じゃあ，今月はこれで終わりにしましょう。来月はというと，まだぜ～んぜん決めていません。借用和音の続きか，非和声音の続きで偶成和音でもやろうかな～，とも思ってますが，お上からこれをやれ！　っていわれればそうするでしょう。それよりも……おハガキの力は偉大です。はい。なんかお手紙ちょうだい。

それではまた来月。

楽譜1　ゼクセクス ステージ1

3
4
B 転
5
B 原

＊この楽譜はコナミから提供されたものではありません

Ⓒ1991 KONAMI ALL RIGHTS RESERVED.

スプライトを使いこなす

Murata Toshiyuki 村田 敏幸

今回はスプライトのお話です。スプライトはいくつかの小さなグラフィックパターンを単独で表示したり移動させたりするものです。シューティングゲームなどでお馴染みでしょう。これからゲームを作りたいと思っている方，ぜひ頑張ってついてきてください。

今月はスプライトを扱う。スプライトといえば，やはりゲーム。というわけで，今回はゲーム，とくにリアルタイムゲームへの応用を気持ち意識して話を進めてみたい。なお，“割り込み”の知識が前提になる箇所があるので，そこらあたりに不安のある人はこの際だから復習しておくことを勧める。

スプライトとBG

一般に，グラフィック画面などの1枚の大きなビットマップ上にグラフィックパターンを表示するときには，座標から算出したVRAMアドレスにパターンデータをどかどかと書き込む。で，移動するときには，せっかく描いたパターンをいったん消してから新たな位置に再描画するわけだ。それだけでも結構な手間なのに，多数のパターンを同時に表示・移動する場合にはパターン同士の重なり合いを考慮しなければならず，手間はさらに増える。描画に先立って描画先の画面内容を取り込んで覚えておき，パターンの余白部分はVRAMに書き込まないよう注意しながら描画して，消すときにはさっき覚えておいた元の画面内容を書き戻す，というような手順を踏むことになるだろう。パターンを単純に表示するだけならともかく，“動かす”という点では，ビットマップ画面は必ずしも有利とはいえないことがわかると思う。

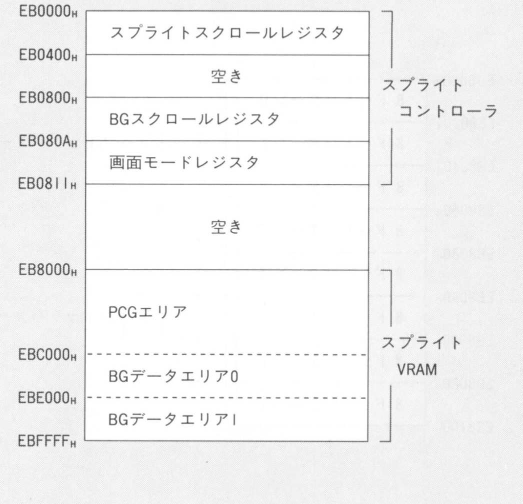
そこで，スプライトの出番となる。スプライトは，比較的小さな複数のグラフィックパターンを独立して表示・移動するための専用ハードウェアだ。あらかじめパターンを定義しておけば，あとはパターンの番号と表示座標などの数バイトのデータを与えるだけで，画面上の任意のドット位置にパターンを表示することができる。座標をちょっと変えてやればパターンは瞬時にして移動するし，パターン番号を順次切り替えればアニメーションするし，複数のパターンが重なった場合の優先順位つきの表示もハードウェアが面倒を見てくれる。この種の特化したハードウェアの常として，表示できるパターンの大きさや数などに制限があったりはするものの，いくつものパターンを高速に動かす必要のあるリアルタイムゲームを作成するうえで，スプライトはいまや欠かせないアイテムだ。

さて，スプライトは，よくバックグラウンド（以下BG）画面というやつとセットになっている。BGはスプライトパターンをマス目状に敷き詰めた格好の画面だ。画面が適当な大きさで縦横に等分されていて，各マスに対応するメモリにパターン番号そのほかのちょっとしたデータを書き込むだけで，あらかじめ定義しておいたパターンが該当位置に表示される，という仕組みだ。ゲームの背景はいくつかの小パターンを組み合わせて構成される場合が多く，BG画面はその名前のとおり，ゲームなどの背景に適している。

X68000のスプライト機能

X68000ではスプライトとBGをシャープカスタムのスプライトコントローラで制御している。流行りの拡大・縮小や回転といった凝った機能はないもの

図1　スプライト関連I/Oのメモリマップ



の，16×16ドット16色のスプライトを128個同時表示（ただし，1水平線上には32個まで)，64×64パターンのBGを最大2面という仕様は，十分高機能といえるだろう。

では，さっそく，スプライトコントローラと関連ハードを紹介していくことにする。

●概要

図1にスプライト関係のメモリマップを示す。EB0000$_H$～EB0811$_H$がスプライトコントローラのレジスタ，EB8000$_H$～EBFFFF$_H$がパターンデータ格納用兼BGの実画面として使われるスプライトVRAMだ。スプライトコントローラのレジスタは，スプライトの制御をするスプライトスクロールレジスタと，BGの制御をするBGスクロールレジスタ，および，画面モードレジスタに分けられる。また，図1以外では，E82200$_H$～E823FF$_H$にスプライトのパレットがある。

もともとスプライトやBGは，少ない手間で大きな実入りを狙ったハードウェアなので，扱いは比較的簡単だ。基本的には，適切なアドレスに適切なデータを書き込むだけといってよい。X68000のI/Oには読み込み専用や書き込み専用のものもあるが，スプライト関係についてはどこも読み書き可能になっており，メモリに対する操作がそのまま適用できる。

ただ，いわずもがな，X68000のI/Oはスーパーバイザ空間に割り付けられているので，アクセス時にはスーパーバイザモードでなければならない。ユーザーモードでアクセスしてもバスエラーが発生するだけだ。また，スプライトコントローラとスプライトVRAMへのアクセス時には“スプライトが表示できる画面モード（表示画面が256×256，512×512ドットモードなど)”になっている必要がある。知ってのとおり，X68000のスプライトは表示画面768×512ドットなどのモードでは使えないわけだが，凶悪なことに“スプライトが使えない画面モードでは，スプライト関連I/Oはメモリ空間から切り離される”のだった[1)]。何もないところにアクセスしようとすれば，やはりバスエラーの洗礼を受けることになる。

細かなことでは，スプライトコントローラとスプライトVRAMはバイト単位でのアクセスが禁止されている点にも注意したい。現実にはバイト単位でアクセスしても問題になることは滅多にないようだが，メーカーはそのような使い方を保証していない。

●スプライトパレット

E82200$_H$～E823FF$_H$には，65536色中16色のスプライトパレットが16組並ぶ(最初のE82200$_H$～E8221F$_H$の1組はテキストパレットと兼用される)。X68000のスプライト/BGは各パターンごとに，これらの16組のパレットテーブルのうちのどれか1組を割り当てることができる。このため，X68000のスプライト/BGの色に関する表示能力は，

1パターンにつき65536色中16色

1画面につき65536色中256（＝16×16）色

となる。もっとも，各パレットテーブルとも，パレットコード0は透明色（カラーコード0）にしておくことになるから，実際に同時に表示できる1画面あたりの色数はもう少し減る。

なお，IOCS SP_INITでスプライト周りを初期化すると，スプライトパレット1～15が16色モードのグラフィックパレットと同じ色構成に初期化されることを付け加えておこう。

●スプライトVRAM

EB8000$_H$～EBFFFF$_H$にマップされた8000$_H$バイトのスプライトVRAMはスプライト/BGのパターン定義用のPCGエリアと，BGの実画面であるBGデータエリアに使われる。すべてをPCGエリアとして使用した場合，16×16ドット16色のパターンを256個定義できるだけの容量があるわけだ。ただし，スプライトVRAM後半4000$_H$バイトは2面分のBGデータエリアと兼用されるため，BGを使うと定義できるパターン数は最悪半分までに制限される。むしろ，標準のPCGエリアはEB8000$_H$～EBBFFF$_H$までで，

1） ただし，スプライトコントローラの画面モードレジスタとスプライトパレットは画面モードによらずアクセス可能。

2） BGデータエリアは2画面分しかないから，0と1の1ビットで区別できるはずだが，仕様では必ず00と01の2ビットで指定するよう決められている。第10ビットの拡張用ビットの存在といい，将来の拡張を見越しているのだろうか。

図2　8×8ドットBGパターンの構造

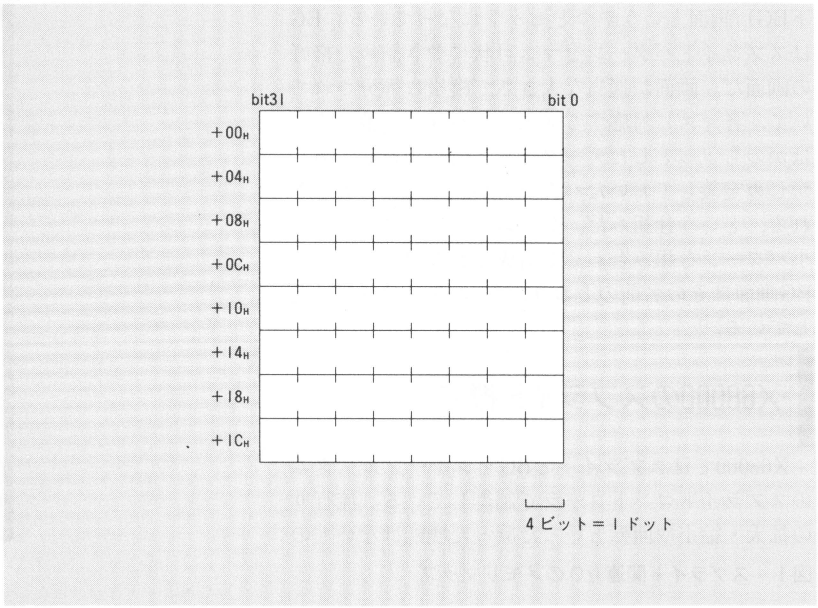
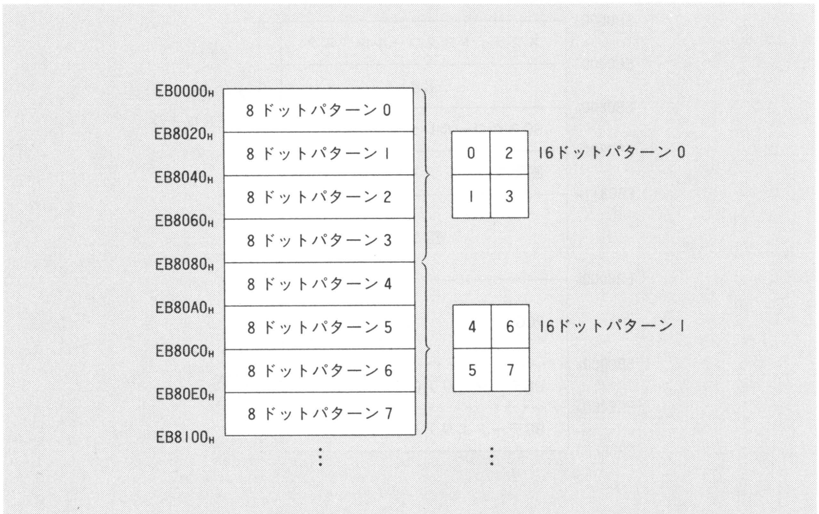


図3　16×16ドットパターンと8×8ドットパターンの対応



BGデータエリアを1面しか使わなければ余ったもう1面分のメモリ，BGをまったく使わなければ2面分のメモリがパターン定義用に流用できると考えたほうがいいようだ。

さて，X68000のスプライト/BGが扱うパターンには16×16ドットのものと，8×8ドットのものの2種類がある。8×8ドットのほうは，表示画面256×256ドットモード時のBGパターンとして使われるのみだが，PCGエリアはこの小さいほうのパターンを基準に構成されている。図2にPCGエリア上での8×8ドットのパターンのドット構成を示した。1ドットが4ビットで，左側のドットが上位ビットに対応するよう，横8ドットが1ロングワードにまとめられ，これが8ライン分で8×8ドットパターンを形作る。16×16ドットパターンのほうは，図3のように4つの8×8ドットパターンを組み合わせた形で表される。

各パターンにはアドレスの小さい順に通し番号が振られ，スプライトスクロールレジスタやBGデータエリアでは，この番号でどのパターンを使うかを指定する。16×16ドットパターンの場合，パターン番号は0～127だ。8×8ドットパターンの場合は0～511となるわけだが，パターン番号は8ビットで指定する仕様になっているため，実際には0～255のみが有効となる。

BGデータエリアは1面あたり64×64パターンの大きさを持ち，1ワードと1パターンが対応する（図4）。8ビットのパターン番号と4ビットのパレットテーブル番号，あと上下/左右の反転の有無を各1ビットで設定すればBG上にパターンが1個表示される。1個のBGパターンの大きさは，表示画面512×512ドットモード時は16×16，256×256ドットモード時は8×8と切り替わるので，それに合わせて16×16ドットのパターン番号と8×8ドットのパターン番号を使い分ける必要がある点にだけ気をつけよう。

●BGスクロールレジスタ

EB0800$_H$～EB0809$_H$には，BG2画面の表示開始座標を指定する2組のレジスタと，BGの表示関係の制御をするBGコントロールレジスタが並ぶ（図5）。表示開始座標を指定する2組4本のレジスタについてはとくに問題はあるまい。表示画面512×512ドットモード時は0～1023，256×256ドットモード時は0～511の座標値を与えれば表示開始位置が変更される。注意らしい注意といえば，表示画面512×512ドットモード時はBGは1面しか表示されないので2組目のBGスクロールレジスタは意味をなさないことと，2画面が同時表示できる表示画面256×256ドットモード時にはBG0のほうが常に優先的に表示されることぐらいだ。

BGコントロールレジスタは，BG2面の表示/非表示の制御，および，BG（の表示画面）とBGデータエリア（BGの実画面）の対応づけを行うためのものだ[2]。2面のBGに同一のBGデータエリアを割り当てることも許されているので，1画面分のBGデータエリアを縦あるいは横に2分割して使い，重ね合わせ表示するといったこともできる。

BGコントロールレジスタの第9ビットは，一見スプライトとBGの表示/非表示を制御するだけに見える[3]が，実は少し深い意味がある。グラフィックにしろ，テキストやスプライトにしろ，画面表示というものは画面表示関係の回路がCRTの走査とタイミングを合わせてVRAMを読み出して送りつけることで実現されるわけだ。つまり，VRAM（や表示関係デバイスのレジスタ）はCPUからのアクセスとは

3) スプライト/BGの表示/非表示のメインスイッチはビデオコントローラのレジスタ3（E82600$_H$からの1ワード）の第6ビットにあり，このビットと，BGコントロールレジスタの第9ビットがともに1の場合にスプライト/BGは表示される。

図4　BGデータエリアの構造

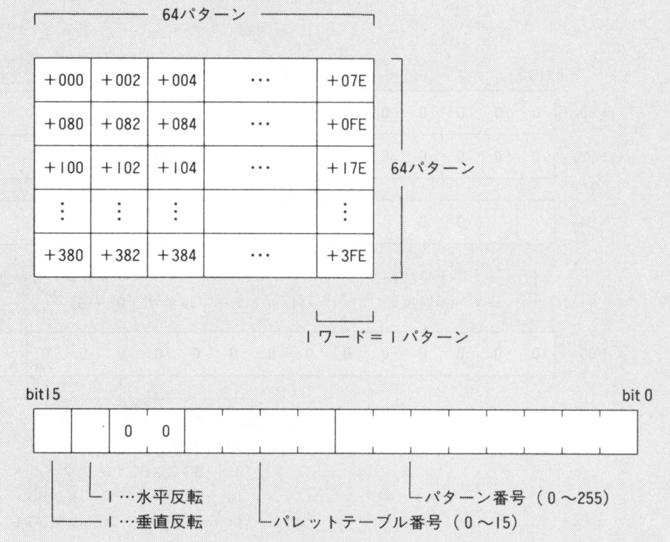
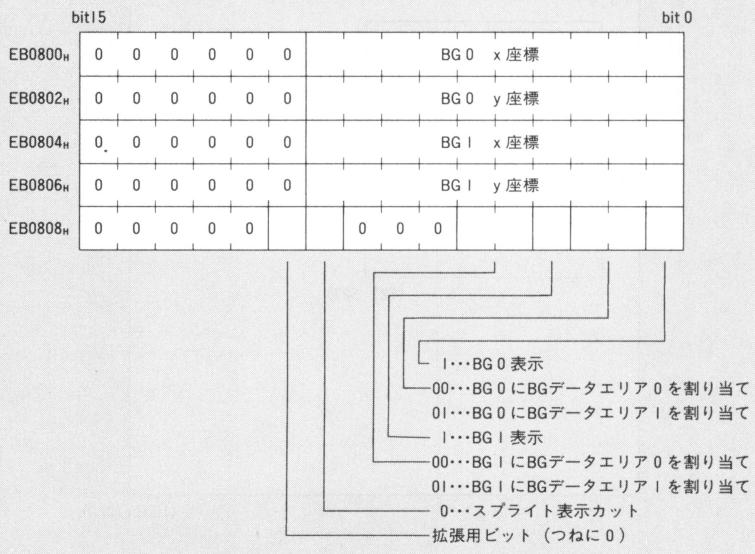


図5　BGスクロールレジスタ



別に，表示用に頻繁に読み出される。通常のメモリは同時に2カ所からアクセスできるようにはできていないので，このような動作を実現するためにはそれなりの細工が必要だ。X68000の場合，テキストVRAMとグラフィックVRAMについてはデュアルポート（読み書き口が2系統ある）のVRAMを使用して対処しているが，スプライト関係については表示用の読み出しを優先して，一定時間に一度，その合間にCPUからのアクセス時間を設けるような形になっている。それ以外のタイミングでアクセスしようとしても，アクセスできる時間がくるまでCPUが待たされる（ウエイトが入る）のだ。で，BGコントロールレジスタの第9ビットはこの待ち時間をなくすためにある。このビットを0にすると，表示用の読み出しが停止し，全時間がCPUからのアクセスに開放される[4]。

当然，表示用の読み出しを行わないのだから，スプライトとBGは表示されなくなる。せっかく高速にアクセスできても表示されないのでは意味がないように見えるかもしれない。だが，CRTには1画面の走査が完了して，つぎの画面の走査に入るまでにちょっと間がある。垂直帰線期間というヤツだ。この隙ならスプライトの表示をカットしてもなんの影響もない。つまり，垂直帰線期間がきたらすかさずスプライトの表示を消して，スプライト関係I/Oへ手早くアクセスして，垂直帰線期間が終わる前に元に戻せばよい。

ちょっと脱線するが，垂直帰線期間に表示関係I/Oを書き換えることは，ちらつき防止の点でも重要だ。グラフィック画面などのスクロール，パレットの変更，そしてスプライトの移動などは，なまじ設定を変更すると即座に画面に影響が出る[5]ために，CRTの走査のタイミング次第では画面の上側と下側の不一致を引き起こす。画面の真ん中あたりを走査しているときにこれらのハードウェアを操作すると，上半分は設定変更前の画面，下半分が設定変更後の画面という中途半端な状態になり，これが人間の目には不快なちらつきとして映る。このちらつきを防ぐためにも，表示関係I/Oへのアクセスは走査の切れ目である垂直帰線期間に行うことが望ましい。

4)　ただし，スプライト関係I/Oへのアクセスには1ワードあたり4クロック程度（実測値から算出した値）のウエイトが無条件に入るようだ。
5)　厳密には，I/Oを操作してから影響がCRTに現れるまでに水平走査線1〜数本分程度の遅れはあるのがふつうだ。

図6　スプライトスクロールレジスタ

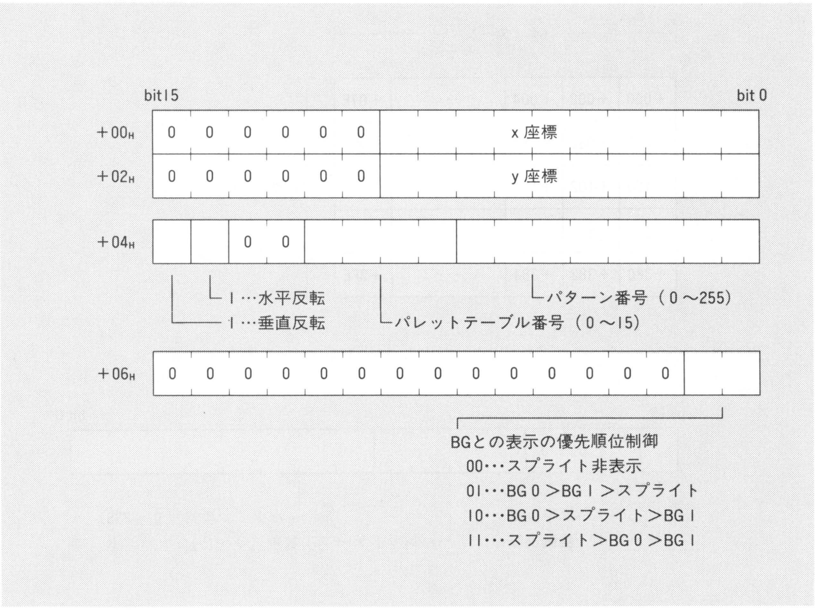
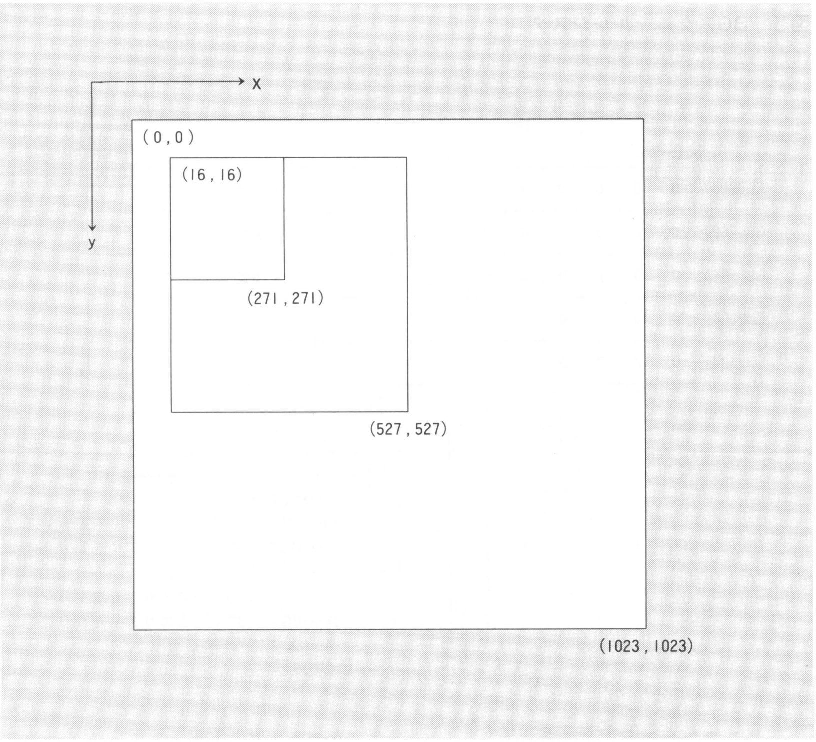


図7　スプライト仮想座標系



●スプライトスクロールレジスタ

EB0000H〜EB03FFHには1ワード×4本のレジスタを1組とする128組のスプライトスクロールレジスタが並ぶ。1組のレジスタがスプライト1個に対応し，レジスタが128組だから，1画面に同時に表示できるスプライトの個数も128個[6]。各スプライトには0からの通し番号がついていて，スプライトどうしが画面上で重なった場合は，番号の若いほうが優先して表示される。

1組あたりのレジスタ構成は図6のようになっている。頭2本のレジスタで指定する座標は下位10ビットのみが有効で，したがって座標値の範囲は0〜1023，スプライト仮想座標系の大きさは1024×1024となる。CRTにはこの仮想座標系のうち，表示画面512×512ドットモード時は(16,16)−(527,527)の範囲が，256×256ドットモード時は（16,16)−(271,271）の範囲が表示される（図7）。表示画面の左上隅にスプライト仮想座標系の（16,16）が対応するというのが少々変則的だが，これは，画面の左端や上端をまたいでスプライトを表示できるようにするための配慮だろう。

●画面モードレジスタ

EB080AH〜EB0811Hの画面モードレジスタはIOCS CRTMODで画面モードを設定するときにCRTCなどと一緒に設定されるので，直接アクセスする機会はあまりないだろう。というわけで，すっ飛ばす。

どうしてもという人は『Inside X68000』などのハードウェア資料を参照してもらいたい。

簡単なサンプル

とりあえず，スプライトを1個動かしてみたのがリスト1だ。実行すると，左上隅に黄色で縁取られた青い四角が表示され，マウスの動きに連動して動く。スプライトによるマウスカーソルだ。何かキーを押すと終了する。

画面モードを表示画面512×512ドットのモードにしてから，カーソルを消し，マウス周りの初期化を行うところまではいいだろう。

続く27～34行がスプライト関係の初期化だ。IOCS SP_INITでスプライトコントローラのレジスタとスプライトVRAMを0クリアし，スプライトパレットを初期化して，SP_DEFCGで74～82行に用意したパターンをパターン番号1に定義してから，SP_ONでスプライトを表示状態にしている。初期化はどうせ1度しか行わないのだからという理由でひたすらIOCSに頼ったが，直接I/Oを操作するように変更するのは簡単だろう。

36～38行はスプライト0のパターン番号，パレットテーブル番号，上下/左右反転の有無，表示の優先順位の設定だ。スプライトスクロールレジスタ0の後半2本のレジスタに，パターン番号は1，パレットテーブル番号は1，反転なし，BG2面よりも優先して表示(もっとも，リスト1ではBGは使っていないが)，というデータを書き込んでいる。たったこれだけの書き込みにいちいちスーパーバイザモードへ移行するのは馬鹿らしいので，IOCS B_LPOKE(任意のアドレスへロングワードデータを書き込む）を使った。

40～42行では垂直帰線期間1回ごとの割り込みを設定している。前述のように，スプライト関係I/Oへのアクセスは垂直帰線期間に行うのが定石であり，リスト1ではこの垂直帰線期間開始の検出に割り込みを使う。

メインルーチンの残りは事実上何もしていない。44～49行のループでキー入力を待ち，何かキーが押されたら，割り込みの設定を解除したり，スプライトを非表示にしたりといった後始末をして終了するだけだ。スプライトは44～49行でキー入力待ちしている間に割り込み処理ルーチンが動かす。

肝心の割り込み処理ルーチンは62～72行にある。ここでは，マウスカーソル位置を取得して[7]，スプライト座標系へ変換し（x，yともに16ドットずつずれている分の補正)，スプライトスクロールレジスタ0に設定している。このプログラムではスプライトを1個しか動かさないので速度の点では問題ないのだが，“型”ということで，スプライトスクロールレジスタへの書き込みの前後ではBGコントロールレジスタの第9ビットを操作してみた。なお，割り込み処理中はスーパーバイザモードなので，I/Oアクセスに先立って走行モードを切り替える必要はない点に注意してほしい。

ゲーム志向のサンプル

リスト1が単純すぎたので，もう1本のサンプルをリスト2に用意した。リスト2は512×512ドットの画面でいい加減な背景が横スクロールする上を，128個のスプライトが連なって円を描いて動くという，意味不明なりにスクロールタイプのゲームの核心に迫るデモだ。多数のスプライトを動かす方法と，背景の（単純な繰り返しではない）スクロールのさせ方はこのプログラムリストを読むとわかる。

スプライトの移動経路と背景のマップデータはX-BASICプログラムで生成するようになっているので，リスト2をアセンブルするときには，先にリスト3，4を実行し，生成されるファイルをカレントディレクトリに置いておくこと。ちなみに，リスト3はスプライトの移動経路の座標テーブル（適当な大きさの円周を400分割した各点の座標を並べてある）を，リスト4は適当な文字列の文字パターンからBGデータエリアにそのままセットできる形式のマップデータ(文字パターンの各ドットを1個のBGパターンに対応させた）を作成するプログラムだ。

では，まず多数のスプライトの動かし方について見ていこう。リスト1では動かすスプライトが1個だけだったのと，マウスカーソルを動かすというプログラムの性格上，スプライトの移動先座標の算出も垂直帰線期間中に行った。しかし，垂直帰線期間は1.数ms[8]しかなく，あまり多くの処理をする時間はない。多数のスプライトを動かす場合，スプライトスクロールレジスタを書き換えるだけでもそれなりに時間がかかるので，移動先の算出は垂直帰線期間外（ということは垂直表示期間）にすませておき，垂直帰線期間は物理的なI/Oアクセスに専念するべきだ。つまり，個々のスプライトのつぎの表示座標を算出したら，それを直接スプライトコントローラへ設定せずに，いったんメインメモリ側のバッファに溜めておいて，データが全部揃ってから，つぎの垂直帰線期間でごっそりと転送するのだ。バッファの大きさ・構造をスプライトスクロールレジスタと揃えておけば，垂直帰線期間中の仕事は単純なブロック転送になる。

というあたりを頭に入れてもらったところで，リスト2中のスプライトの移動周りを拾い読みしてみる。36～42行でバッファ（リスト中のコメントでは仮想スプライトスクロールレジスタと呼んでいる）を初期化する。飛んで51～53行で割り込みを設定するのだが，その直前の49行で内部的なフラグの設定を行っている。このフラグはメインルーチンと割り込み処理ルーチン間の同期をとるためのものだ。バッファにデータが揃ったらメインルーチンはこのフラグを立てる。割り込み処理ルーチン側では，割り込みがかかるたびにこのフラグを調べて，寝ていた

6) スプライトスクロールレジスタの直後のメモリ空間は“もう128組分のレジスタが収まるぐらい”空いているから，将来に期待してみるのもいいかもしれない（無責任)。

7) 前にも話したと思うが，本来，割り込み処理ルーチン内でシステムコールを発行するのはあまりよいことではない。割り込みが発生した時点でシステムコールの処理中だったりすると誤動作する危険がある（X68000のシステムコールは基本的にこのような呼び出しができるようにはなっていない)。ただ，マウスカーソルを取得するIOCS MS_CURGTはIOCSの内部ワークを読み出すだけのシンプルさなので，実害はないはずだ。

8) スプライトの書き換えに使える時間は，高解像度モード時約1.7ms，標準解像度モード時約1.4ms。

らまだデータが揃っていないものと判断して何もせずに抜ける。フラグが立っていたらバッファからスプライトスクロールレジスタへのデータ転送を行って，フラグを寝かせる。メインルーチン側はフラグが寝たことで割り込み処理が完了したことを知り，つぎの移動先座標の算出に取りかかる。このようなフラグを設けないと，バッファにデータが揃っていないのにスプライトスクロールレジスタが中途半端に更新されてしまったり，逆にまだスプライトスクロールレジスタにデータが転送されないうちに，バッファを更新してしまったりといったことが起こる。

個々のスプライトの移動先の座標を算出するのが63～67行のループだ。もっとも，リスト２ではあらかじめ移動経路の座標がテーブルになっているので，ここではテーブルの適当な位置から３つおきに座標データを引いているだけ。説明するまでもないとは思うが，このテーブル参照開始位置をひとつずつずらすことで，各スプライトが連なって動くという動作を実現している。

バッファへの座標設定が終わったら69行でフラグを立て，割り込み処理ルーチンの活動を再開する。するとつぎに割り込みがかかったときに116～120行でバッファ内容が実際のスプライトスクロールレジスタに転送される。リスト２ではブロック転送をロングワード単位でのmoveのループで行っているが，速度を稼ぎたかったら，ループを展開するか，DMAを使うかすることになる。

転送がすんだかどうかを調べているのが83～84行だ。69行で負の値（－１）を書き込んだメモリが正の値になるまで待つという，奇妙な格好になっている。割り込みがかかったときに割り込み処理ルーチンがこのメモリを書き換えることを知らなければ，ただの無限ループに見えるだろう。

リスト２のもうひとつのポイントである背景のスクロール方法は，ポイントというわりにはなんということはない。右から左への横スクロールの場合，少しスクロールしては，表示画面のすぐ右側（つぎにスクロールすると表示画面に入ってくる部分）を更新すればよいだけのことだ。グラフィック画面を使う場合は１ドットスクロールするたびに縦１ライン分を書き換えることになるし，BGを使う場合は８ドット（表示画面256×256ドットモード時）または16ドット（同512×512ドットモード時）スクロールするごとに縦32パターンを書き換えることになる[9]。

リスト２では，つぎに使うマップデータへのポインタと，つぎに書き換えるBGデータエリア上のアドレスをワークに格納しておき，垂直帰線期間での割り込み処理ルーチンで毎回１ドットスクロールし，16回スクロールしたらワークから２つのポインタを取り出して32パターン分の転送を行う（128～133行）ようになっている。プログラムを読むうえでは，BGデータエリア端でのつじつま合わせ（90～92行）と，16を数えるカウンタがデクリメント式ではなくローテート式になっている点[10]に注意してもらいたい。

というあたりで，一応，スプライトの使い方の基本的な型は示せたようだ。今回の話程度のことがわかっていれば，そこそこ派手なゲームも作れるだろうし，手始めにスプライトがいっぱい乱れ飛ぶだけのデモあるいは環境プログラムなんか作ってみるのもいいかもしれない。

9)　垂直帰線期間での処理時間を一定にしたければ，１ドットスクロールするごとに２あるいは４パターンずつ書き換えていくという方法もとれるだろう。

10)　16ビットデータは16回ローテートすると元に戻る。だから，１ビットだけが立ったデータをローテートしてccrの変化に注目すれば，16を数える“リセット不要の”カウンタになる。

リスト１　SPMOUSE.S

```
*           スプライトを使ったマウスカーソル

            .include        doscall.mac
            .include        iocscall.mac
*
SPSCRLREG           equu        $eb0000 *スプライトスクロールレジスタ
BGCTRLREG           equ         $eb0808 *BGコントロールレジスタ
*
            .text
            .even
*
ent:
            lea.l    inisp(pc),sp

            moveq.l #0,d1                *512×512，16色4画面，高解像度
            IOCS     _CRTMOD             *

            IOCS     _OS_CUROF           *カーソル非表示

            IOCS     _MS_INIT            *マウス初期化
            moveq.l #0,d1                *
            IOCS     _SKEY_MOD           *
            moveq.l #0,d1                *
            move.l   #$01ff_01ff,d2      *
            IOCS     _MS_LIMIT           *

            IOCS     _SP_INIT            *スプライト初期化

            moveq.l #1,d1                *スプライトパターンをひとつ定義
            moveq.l #1,d2                *
            lea.l    pat(pc),a1          *
            IOCS     _SP_DEFCG           *

            IOCS     _SP_ON              *スプライト表示

            lea.l    SPSCRLREG+4,a1      *スプライト0のパターン番号他を設定
            move.l   #$01_01_0003,d1     *
            IOCS     _B_LPOKE            *

            moveq.l #1,d1                *垂直帰線期間ごとの割り込みを設定
            lea.l    intent(pc),a1       *
            IOCS     _VDISPST            *

loop:       IOCS     _B_KEYSNS           *キーが押されたら終了する
            tst.l    d0                  *
            beq      loop                *
            IOCS     _B_KEYINP           *
            tst.b    d0                  *
            beq      loop                *

done:       suba.l   a1,a1               *垂直帰線期間での割り込みを禁止
            IOCS     _VDISPST            *

            IOCS     _SP_OFF             *スプライト非表示

            moveq.l #-1,d1               *ソフトウェアキーボード許可
            IOCS     _SKEY_MOD           *
            IOCS     _OS_CURON           *カーソル表示

            DOS      _EXIT
*
intent:
            move.l   d0,-(sp)
            move.w   #$000,BGCTRLREG *スプライト表示カット

            IOCS     _MS_CURGT           *マウスカーソル位置取得
            addi.l   #$0010_0010,d0      *スプライト座標系へ変換
            move.l   d0,SPSCRLREG        *スプライト移動

            move.w   #$200,BGCTRLREG *スプライト表示
            move.l   (sp)+,d0
            rte
*
pat:                                     *適当なスプライトパターン
            .dc.l    $dddddddd,$d3333333,$d3333333,$d3333333
            .dc.l    $d3333333,$d3333333,$d3333333,$d3333333
            .dc.l    $d3333333,$d3333333,$d3333333,$d3333333
            .dc.l    $d3333333,$d3333333,$d3333333,$dddddddd
            .dc.l    $dddddddd,$3333333d,$3333333d,$3333333d
            .dc.l    $3333333d,$3333333d,$3333333d,$3333333d
            .dc.l    $3333333d,$3333333d,$3333333d,$3333333d
            .dc.l    $3333333d,$3333333d,$3333333d,$dddddddd
*
            .stack
            .even
*
            .ds.l    256
inisp:

            .end
```

リスト2　GURU.S

```
*               スプライトとBGのデモ
                .include        doscall.mac
                .include        iocscall.mac
*
NMAXSP          equ     128     *スプライトの最大数
SPPALET         equu    $e82200 *スプライトパレット
SPSCRLREG       equ     $eb0000 *スプライトスクロールレジスタ
BGSCRLREG0      equ     $eb0800 *BGスクロールレジスタ0
BGSCRLREG1      equ     $eb0804 *BGスクロールレジスタ1
BGCTRLREG       equ     $eb0808 *BGコントロールレジスタ
PCGAREA         equ     $eb8000 *スプライトパターン領域
BGDATAAREA0     equ     $ebc000 *BGのBGデータエリア0
BGDATAAREA1     equ     $ebe000 *BGのBGデータエリア1
*
                .text
                .even
*
ent:
        lea.l   inisp(pc),sp
        moveq.l #0,d1                   *512×512, 16色4画面, 高解像度
        IOCS    _CRTMOD                 *
        IOCS    _OS_CUROF               *カーソル非表示
        IOCS    _SP_INIT                *スプライト初期化
        moveq.l #1,d1                   *スプライトパターンを
        moveq.l #1,d2                   *  ひとつ定義
        lea.l   pat(pc),a1              *
        IOCS    _SP_DEFCG               *
        IOCS    _SP_ON                  *スプライト表示
        lea.l   spreg(pc),a6            *仮想スプライトスクロールレジスタを
        moveq.l #0,d0                   *  初期化
        move.l  #$01_01_0003,d1         *
        moveq.l #NMAXSP-1,d2            *
inilp:  move.l  d0,(a6)+                *
        move.l  d1,(a6)+                *
        dbra    d2,inilp                *
        move.w  d0,bgx                  *BGの表示x座標を初期化
        move.w  #$4000,bgctr            *'16'を数えるカウンタを初期化
        move.l  #bgdat,bgsour           *マップデータとBGデータエリアへの
        move.l  #BGDATAAREA1+32*2,bgdest        *ポインタを初期化
        sf.b    reqflg                  *割り込み処理停止
        moveq.l #1,d1                   *垂直帰線期間ごとの割り込みを設定
        lea.l   intent(pc),a1           *
        IOCS    _VDISPST                *
        lea.l   spdat(pc),a0            *a0 = スプライト座標データ先頭
        lea.l   spdate(pc),a1           *a1 = 同末尾
        lea.l   bgdat(pc),a2            *a2 = BGマップデータ先頭
        lea.l   bgdate(pc),a3           *a3 = 同末尾
loop:   movea.l a0,a4
        lea.l   spreg(pc),a5            *a5 = 仮想スプライトスクロールレジスタ
        moveq.l #NMAXSP-1,d0            *スプライトを移動
setlp:  move.l  (a4)+,(a5)+             *(仮想スプライトスクロールレジスタ上)
        addq.l  #8,a4                   *
        addq.l  #4,a5                   *
        dbra    d0,setlp                *
update: st.b    reqflg                  *スプライトスクロールレジスタの転送開始
        IOCS    _B_KEYSNS               *キーが押されたら終了する
        tst.l   d0                      *
        beq     nokey                   *
        IOCS    _B_KEYINP               *
        tst.b   d0                      *
        bne     done                    *
nokey:  subq.l  #4,a0                   *座標データへのポインタを進める
        cmpa.l  a0,a1                   *ポインタがデータ末に達したら
        bcs     wait                    *
        lea.l   spdat(pc),a0            *  ポインタをリセット
wait:   tst.b   reqflg                  *スプライトスクロールレジスタへの
        bmi     wait                    *  転送が終わるのを待つ
        addq.w  #1,bgx                  *BGの表示x座標を更新
        tst.w   bgctr                   *いまBGを書き換えたばかりなら
        bpl     loop                    *
        addq.b  #2,bgdest+3             *  BGデータエリアへのポインタを更新
        bpl     skip                    *
        sf.b    bgdest+3                *
skip:   cmpa.l  bgsour,a3               *  マップデータへのポインタ
        bhi     loop                    *
        move.l  a2,bgsour               *
        bra     loop                    *繰り返す
done:   suba.l  a1,a1                   *垂直帰線期間での割り込みを禁止
        IOCS    _VDISPST                *
        IOCS    _SP_OFF                 *スプライト非表示
        IOCS    _OS_CURON               *カーソル表示
        DOS     _EXIT
*
intent:
        bclr.b  #7,reqflg               *転送すべきデータが揃っていなければ
        beq     retn                    *  何もせずに戻る
        movem.l d0/a0-a1,-(sp)
        move.w  #$000,BGCTRLREG         *スプライト表示カット
        lea.l   spreg(pc),a0            *仮想スプライトスクロールレジスタから
        lea.l   SPSCRLREG,a1            *(実)スプライトスクロールレジスタへ
        move.w  #NMAXSP*2-1,d0          *  ブロック転送
cpylp:  move.l  (a0)+,(a1)+             *
        dbra    d0,cpylp                *
        move.w  (a0)+,$400(a1)          *BG表示x座標更新
        rol.w   (a0)+                   *カウントタウン
        bpl     nset                    *16回スクロールしていなければ
                                        *  BGの書き直しは不要
        movea.l (a0)+,a1                *a1 = 書き換えるBGデータエリア位置
        movea.l (a0),a0                 *a0 = マップデータ
        moveq.l #32-1,d0                *縦32キャラクタ分転送
bglp:   move.w  (a0)+,(a1)              *
        lea.l   64*2(a1),a1             *
        dbra    d0,bglp                 *
        move.l  a0,bgsour               *マップデータへのポインタを更新
nset:   move.w  #$203,BGCTRLREG         *スプライト表示
        movem.l (sp)+,d0/a0-a1
retn:   rte
*
pat:                                    *適当なスプライトパターン
        .dc.l   $dddddddd,$d3333333,$d3333333,$d3333333
        .dc.l   $d3333333,$d3333333,$d3333333,$d3333333
        .dc.l   $d3333333,$d3333333,$d3333333,$d3333333
        .dc.l   $d3333333,$d3333333,$d3333333,$dddddddd
        .dc.l   $dddddddd,$3333333d,$3333333d,$3333333d
        .dc.l   $3333333d,$3333333d,$3333333d,$3333333d
        .dc.l   $3333333d,$3333333d,$3333333d,$3333333d
        .dc.l   $3333333d,$3333333d,$3333333d,$dddddddd
*
spdate: .include        spdat.inc       *スプライト座標データ
spdat:  .include        spdat.inc       *
bgdat:  .include        bgdat.inc       *マップデータ
bgdate:
*
        .bss
        .even
*
spreg:  .ds.w   NMAXSP*4                *仮想スプライトスクロールレジスタ
bgx:    .ds.w   1                       *BGのx表示座標
bgctr:  .ds.w   1                       *BG更新用カウンタ
bgdest: .ds.l   1                       *つぎに書き換えるBGデータエリア位置
bgsour: .ds.l   1                       *つぎに書き込むマップデータ位置
reqflg: .ds.b   1                       *スプライトスクロールレジスタを
                                        *  更新する必要があるかのフラグ
                                        *(bit7=1 ... 未更新)
*
        .stack
        .even
*
        .ds.l   256
inisp:
        .end
```

リスト3　SPDAT.BAS

```
 10 int fp,i,j,x,y
 20 str s
 30 fp=fopen("spdat.inc","c")
 40 for i=0 to 511
 50     x=256+int(160*cos(pi()*i/256))+16
 60     y=256+int(160*sin(pi()*i/256))*4/3+16
 70     s=" .dc.w "+itoa(x)+","+itoa(y)+chr$(13)+chr$(10)
 80     fwrites(s,fp)
 90 next
100 fclose(fp)
```

リスト4　BGDAT.BAS

```
 10 int fp,x,y,n
 20 str s="魑魅魍魎",t[256]
 30 screen 1,0,1,1
 40 n=strlen(s)*12
 50 symbol(0,4,s,1,1,2,1,0)
 60 fp=fopen("bgdat.inc","c")
 70 for x=0 to n-1
 80 t=chr$(9)+".dc.w"+chr$(9)
 90 for y=0 to 30
100     t=t+itoa(256+point(x,y))+","
110 next
120 t=t+itoa(256+point(x,y))+chr$(13)+chr$(10)
130 fwrites(t,fp)
140 line(x,0,x,31,0)
150 next
160 fclose(fp)
```

Oh!X LIVE in '92

X68000・Z-MUSIC/PCM8.X用 ドラゴンセイバーより **氷穴**
©NAMCO ALL RIGHTS RESERVED
Ueda Hiroshi 上田 浩司

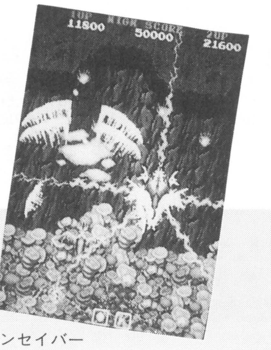
X68000用 **ガラガラヘビがやってくる**
Nezu Shinya 祢津 伸也

X1/X1turbo用 出たな!! ツインビーより **風の贈り物**
©KONAMI
Nagasaka Kazuhiko 長坂 和彦

今月はX68000用が2本と，X1用が1本です。特にドラゴンセイバーはリストが長めですので，夏休みでヒマを持て余している方はぜひ打ち込んでみてください。暑さとダルさにまいってる人にまではおススメしませんけどね。

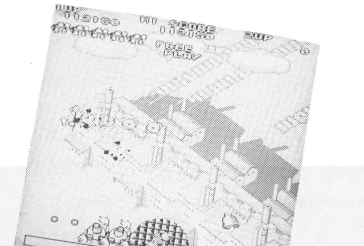
28チャンネル使用中!

今月の1曲目はナムコのアーケードゲーム，ドラゴンセイバーから6面の曲「氷穴」をお届けしましょう。Z-MUSICシステム用で，CM-64相当品が必要です。さらにPCM8.Xを必要としますので，組み込んでおいてください。



ドラゴンセイバー

ガラガラヘビがやってくる


出たな!! ツインビー

冒頭からちょっと贅沢なシステム構成を必要とする曲になってしまいましたが，曲のデキも相応のものとなっていますので，納得してください。

さて，ミキシングについてですが，X68000本体のボリュームは12時方向，CM-32Pは3時方向，MT-32はVOL80(オーディオミキシングケーブル使用の場合)にしてみてください。ミキサーなどで自分の好みに合わせてミキシングしてもよいでしょう。

上田君はこの曲を4度も作っているようですね。おそらく，作り直すたびにレベルが上がったのではないでしょうか。音取りやコーディングに普段の4倍も時間をかけて作ったそうです。時間をかければよいというわけではないでしょうが，今回はちゃんと実を結びましたね。

非常に長いプログラムで打ち込みがいもありますが，きっと入力した努力はむくわれると思います。

最後に，トータルカウンタについてですが，Z-MUSICシステムが最新版でないとエラーになってしまうかもしれません。NETや友人などを通じて最新版を手に入れるようにしましょう。

とんねるちゃんがやってきた!

さて，豪華なシステムが必要だった1曲目とは対照的に，X68000ユーザーならば誰でも聴ける曲もお届けしましょう。「ガラガラヘビがやってくる」は内蔵FM音源しか使っていません。もちろん，X-BASIC用です。

この「ガラガラヘビがやってくる」はTV番組の「とんねるずのみなさんのおかげです」のオープニングテーマだそうです。作者の祢津君によると「ご存じのとおり～」となっていますが，恥ずかしながら私は知りませんでした。番組名は知っていましたが，とんねるずが大嫌いな私としては見たことがなかったのです。ちなみになぜか曲だけはちゃんと知っていましたが。

このテの曲をLIVE inのコーナーに送ると，イロモノ扱いになることをご存じでしょうか。イロモノになると，作品のデキは2の次，3の次となり，ネタの面白さや新鮮さ，季節ものやタイムリーな選曲が最優先されることになります。もし，あなたがMMLの初心者で，どうしてもこのページに掲載されたいとするならば，こっちの路線を狙うのがてっとり早いと思いますよ。

この作品では「現在の自分の知識をフルに使った」と作者の祢津君は申しております。確かにコンピュータ歴6カ月では限度というものもあるでしょうが，選曲がよかったことも手伝って，非常にうまくまとまっています。コミカルな曲の軽さがちゃんと出ているので立派といえるでしょう。「サンプリングを使いたかった」と手紙にありますが，逆にサンプリングを使うと，重みが増したぶんだけまとめ方が難しくなると思います。

出たよ!! ツインビー

今月のトリをつとめていただく曲はX1用の「風の贈り物」です。Music BASIC用で，PSGを使っている関係でミキシングが必要です。X1turboZより前のユーザーは好みのミキシングで聴いてみてください。

この曲は，出たな!! ツインビーの1面の曲です。X68000ユーザーにはもうすっかり

お馴染みでしょう。製品としてもかなり売れたほうですし，Z-MUSICのディスクにもサンプルでMIDIに対応した曲が入っていましたからね。

さすがにMIDIと比べるのはかわいそうですが，曲のデキとしてはなかなかのものです。FM音源8音＋PSG 3音の構成としては，ピカイチの部類に入るといっても過言ではないでしょう。リストも短めですので，X1ユーザーは絶対に入力しましょう。

曲中に使われているオーケストラヒットの音色ですが，確かに長坂君イチ押しという感じでよくできています。ただし，ボリュームが弱いためかタイミングがずれているように聞こえてしまいます。ちょっと残念。X1ユーザー以外でもFM音源のオケヒットで悩んでいる人は一見（一聴？）の価値はあると思いますよ。

さて，長坂君はちょっと変わったシステムをローダに使っているようですね。基本的には曲をセレクトして演奏させるというセレクタといった雰囲気のものですが，できればランダムプレイやジュークボックスなどの機能を追加したほうが面白いのではないでしょうか。

さらに，1画面あたりに出ている曲数が少ないことも難点といえるでしょう。こちらのほうの改良も手掛けてみてはどうでしょう。完成したらLIVE in宛に投稿してみてくださいね。　　　　(S.K.)

リスト1　ドラゴンセイバー

```
.comment -Dragon Saber-  Stage 6 (C)1990 NAMCO LIMITED

/ composer   Hosoe Shinji.
/ programmer h.U 5/23
/ for ZMUSIC.X + PCM8
/ MIDI MODULE : CM-64
/

/*****************************************************
/ T R A C K   S E T   U P

(i)
(b0)              / Base Channel = Internal


(m1,2000)(aFm1,1)
(m2,2000)(aFm2,2)
(m3,2000)(aFm3,3)
(m4,2000)(aFm4,4)
(m5,2000)(aFm5,5)
(m6,2000)(aFm6,6)
(m7,2000)(aFm7,7)
(m8,2000)(aFm8,8)
(m9,2000)(aAdpcm,9)
(m10,2000)(aAdpcm,10)
(m11,2000)(aAdpcm,11)
(m12,2000)(aMidi10,12)
(m13,2000)(aMidi10,13)
(m14,2000)(aMidi10,14)
(m15,2000)(aMidi10,15)
(m16,2000)(aMidi11,16)
(m17,2000)(aMidi2,17)
(m18,2000)(aMidi3,18)
(m19,2000)(aMidi4,19)
(m20,2000)(aMidi5,20)
(m21,2000)(aMidi6,21)
(m22,2000)(aMidi7,22)
(m23,2000)(aMidi12,23)
(m24,2000)(aMidi12,24)
(m25,2000)(aMidi12,25)
(m26,2000)(aMidi13,26)
(m27,2000)(aMidi14,27)
(m28,2000)(aMidi15,28)


/*****************************************************
/ CM64 INIT

.roland_exclusive 16,22={$7F,00,00,00}
.sc55_part_setup 1={17}
.sc55_part_setup 2={17}
.sc55_part_setup 3={17}
.sc55_part_setup 4={17}
.sc55_part_setup 5={17}
.sc55_part_setup 6={17}
.sc55_part_setup 7={17}
.sc55_part_setup 8={17}
.sc55_part_setup 9={17}
.sc55_part_setup 10={17}
.sc55_part_setup 11={17}
.sc55_part_setup 12={17}
.sc55_part_setup 13={17}
.sc55_part_setup 14={17}
.sc55_part_setup 15={17}
.sc55_part_setup 16={17}

/*****************************************************
/ ADPCM DATA SET

.adpcm_block_data=ds.zpd

/*****************************************************
/ OPM DATA SET

/         AR 1DR D2R  RR 1DL  TL  RS MUL DT1 DT2 AME
Bass
(@70,     31, 15,  6,  3,  2, 10,  1,  0,  0,  0,  0
          31, 14,  6,  3,  2, 27,  1,  0,  0,  0,  0
          30, 14,  9,  3,  2, 10,  1,  0,  0,  0,  0
          30,  7,  6,  4,  0,  0,  1,  0,  0,  0,  0
/         AL  FB  OM
           4,  6, 15)

/         AR 1DR D2R  RR 1DL  TL  RS MUL DT1 DT2 AME
Synthesizer 1
(@71,     26,  9,  9,  4, 11, 26,  1,  5,  0,  0,  0
          28,  6,  8,  3,  9, 25,  0,  0,  0,  0,  0
          26,  6,  7,  4,  9, 27,  1,  2,  0,  0,  0
          28,  5,  8,  5, 11,  1,  1,  1,  0,  0,  0
/         AL  FB  OM
           1,  3, 15)

/         AR 1DR D2R  RR 1DL  TL  RS MUL DT1 DT2 AME
Synthesizer 2
(@72,     26,  3,  1,  1,  7, 29,  0,  3,  0,  0,  1
          26,  2,  1,  3,  1, 24,  0,  1,  0,  0,  0
          26,  1,  0,  4,  1, 24,  1,  1,  0,  0,  0
          26,  1,  1,  4,  1,  1,  1,  1,  0,  0,  0
/         AL  FB  OM
           0,  6, 15)

/         AR 1DR D2R  RR 1DL  TL  RS MUL DT1 DT2 AME
Synthesizer 3
(@73,     15,  5,  1,  5,  0, 21,  0,  1,  0,  0,  0
          19,  7,  1,  5,  0,  7,  0,  1,  0,  0,  0
          15,  2,  0,  5,  0, 20,  0,  1,  0,  0,  0
          19,  2,  0,  5,  0,  6,  0,  1,  0,  0,  0
/         AL  FB  OM
           4,  6, 15)

/         AR 1DR D2R  RR 1DL  TL  RS MUL DT1 DT2 AME
Fm Strings
(@74,      8, 11,  0,  4,  0, 25,  1,  2,  3,  0,  0
          10, 16,  0,  4,  0, 30,  1,  4,  1,  0,  0
           8, 11,  0,  4,  0, 35,  1,  6,  7,  0,  0
          10, 16,  0,  4,  0,  0,  1,  4,  0,  0,  0
/         AL  FB  OM
           2,  7, 15)

/         AR 1DR D2R  RR 1DL  TL  RS MUL DT1 DT2 AME
Fm Vib
(@75,     25, 13,  9,  6, 10, 42,  0,  6,  0,  0,  0
          25, 13,  6,  7,  7, 12,  0,  4,  0,  0,  0
          25, 11,  9,  6,  9, 10,  0,  1,  0,  0,  0
          25, 11,  6,  5, 10,  2,  0,  2,  0,  0,  0
/         AL  FB  OM
           6,  6, 15)

/         AR 1DR D2R  RR 1DL  TL  RS MUL DT1 DT2 AME
Synthesizer 4
(@76,     28,  4,  4,  5,  5, 40,  0,  3,  0,  0,  1
          28,  9,  5,  5,  4, 23,  1,  2,  0,  0,  0
          28,  8,  5,  6,  4, 24,  1,  3,  0,  0,  0
          28,  7,  6,  6,  5,  0,  1,  4,  0,  0,  0
/         AL  FB  OM
           3,  6, 15)

/         AR 1DR D2R  RR 1DL  TL  RS MUL DT1 DT2 AME
Synthesizer 5
(@77,     31,  4,  1,  4,  7, 28,  0,  3,  0,  0,  0
          31,  3,  1,  3,  2, 24,  0,  1,  0,  0,  0
          31,  2,  0,  3,  2, 24,  1,  1,  0,  0,  0
          31,  2,  1,  3,  2,  1,  1,  1,  0,  0,  0
/         AL  FB  OM
           0,  7, 15)

/         AR 1DR D2R  RR 1DL  TL  RS MUL DT1 DT2 AME
Synthesizer 6
(@78,     31,  8,  3,  5, 10, 33,  1,  3,  0,  0,  0
          31,  8,  3,  5,  2, 27,  1,  3,  0,  0,  0
          31,  8,  3,  5,  2, 27,  1,  1,  0,  0,  0
          31,  8,  3,  5,  5,  0,  1,  1,  0,  0,  0
/         AL  FB  OM
           1,  2, 15)

/         AR 1DR D2R  RR 1DL  TL  RS MUL DT1 DT2 AME
Synte Bass
(@79,     31, 13,  2,  0,  1,  5,  1,  1,  0,  0,  0
          31, 14,  2,  0,  1, 20,  1,  0,  0,  0,  0
          31, 13,  6,  0,  1,  5,  1,  1,  0,  0,  0
          31,  7,  2,  0,  0,  0,  1,  0,  0,  0,  0
/         AL  FB  OM
           3,  6, 15)

/         AR 1DR D2R  RR 1DL  TL  RS MUL DT1 DT2 AME
Synthesizer 7
(@80,     31,  7,  6,  4,  4, 33,  0,  5,  7,  0,  0
          31,  8,  5,  4, 15,  5,  0,  4,  7,  0,  0
          31,  6,  5,  3,  5,  2,  0,  0,  3,  0,  0
          31,  8,  5,  2,  4,  0,  0,  2,  3,  0,  0
/         AL  FB  OM
           5,  2, 15)

/         AR 1DR D2R  RR 1DL  TL  RS MUL DT1 DT2 AME
Bass2
(@81,     31, 15,  6,  3,  2, 11,  1,  0,  0,  0,  0
          31, 15,  6,  3,  2, 28,  1,  0,  0,  0,  0
          30, 15,  9,  3,  2, 12,  1,  0,  0,  0,  0
          30,  8,  6,  3,  0,  0,  1,  0,  0,  0,  0
/         AL  FB  OM
           4,  7, 15)

```

```
/*******************************************************
/ CM64 System SETUP

/ LA SOUND PART
.roland_exclusive 16,22 ={
                  $10,0,1                    / address
                  1,4,4,                     / reverb
                  4,4,4,4,3,3,0,0,5          / ptl reserve
                  1,2,3,4,5,6,7,8,9}         / MIDI ch#
/ PCM SOUND PART
.roland_exclusive 16,22 ={
                  $52,0,1                    / address
                  2,4,5                      / reverb
                  5,5,5,5,5,5                / ptl reserve
                  10,11,12,13,14,15}         / MIDI ch#

/*******************************************************
/ DRUM SETUP

.mt32_drum_setup 36,16 ={64,85,7,1}          / Bass drum
.mt32_drum_setup 38,16 ={65,85,7,1}          / Snare drum
.mt32_drum_setup 40,16 ={70,77,4,1}          / Close  Hihat
.mt32_drum_setup 41,16 ={71,81,4,1}          / Open   Hihat
.mt32_drum_setup 43,16 ={70,77,10,1}         / Close  Hihat
.mt32_drum_setup 45,16 ={71,81,10,1}         / Open   Hihat
.mt32_drum_setup 48,16 ={72,95,10,0}         / Crash Cymbal
.mt32_drum_setup 50,16 ={72,95,4,0}          / Crash Cymbal

/*******************************************************
/ MML DATA SET

(t1)    t187
(t1)    r4
(t2)    r4
(t3)    r4
(t4)    r4
(t5)    r4
(t6)    r4
(t7)    r4
(t8)    r4
(t9)    r4
(t10)   r4
(t11)   r4
(t12)   r4
(t13)   r4
(t14)   r4
(t15)   r4
(t16)   r4
(t17)   r4
(t18)   r4
(t19)   r4
(t20)   r4
(t21)   r4
(t22)   r4
(t23)   r4
(t24)   r4
(t25)   r4
(t26)   r4
(t27)   r4
(t28)   r4
/
/ F M   B a s s
/
(t1) @70 q8 @k1 l8 p3 v15 o2 r2.
(t1) [do]
(t1) |:2a1&a1 f1&f1 | g1&g1&g1&g1 :|
(t1)                      g1&g1&g1&g2 v16 >q6f<q8f>q6g<q8g v15
(t1) q7 |:2|:4b-rrb-r2:||:4arrar2:|:|
(t1) |:2|:2 a-a-ra- ra-a-a-  a-a-ra- r<a->a-a- :|
(t1)      |:2 g-g-rg- rg-g-g-  g-g-rg- r<g->g-g- :|
(t1)      |:2 e e re  re e e   e e re  r<e >e e  :|
(t1)      |:2 f f rf  rf f f | f f rf  r<f >f f  :| ffrq1f q7r<f>ff
(t1) :|
(t1)      |:8 q8b-4q7b-q8b-| rb-rb-:| r2
(t1) |:2|:4 q8b-4q7b-q8b-  rb-rb-:|
(t1)      |:4 q8a-4q7a-q8a-  ra-ra-:|
(t1)      |:4 q8g-4q7g-q8g-  rg-rg-:|
(t1)      |:4 q8e4 q7e q8e   re re :|
(t1) :| q7
(t1) f (f,b)4&b*20 (b,<e-)4&e-*20 (e-,>a-)4&a-*20 (a-,<d-)4&d-*20 (d-,g-)4&g-*20
(t1) >b- (b-,<e-)4&e-*20 >
(t1) f (f,<c)4&c*20 (c,f)4&f*20 (f,>a)4&a*20 (a,<d)4&d*20 (d,g)4&g*20
(t1) >b (b,<e)4&e*20
/
/ F M   M e l o d y  1
/
(t2) @71 q8 @k00 l8 p3 v14 o4 @m6 @h40 @s390 r2.
(t2) [do]
(t2) |:2
(t2)  a*18&(a,b)6 b*18&(b,<c)6 c*18&(c,g)6 (e,g)20&g*292
(t2)  (g,e)5&e*19&e4 (e,>a)5&a*19&a1 |
(t2)  a*18&(a,<c)6 c*18&(c,g)6 g*18&(g,a)6 a*18&(a,>g)6
(t2)  g*18&(g,a)6 a*18&(a,b)6  b (b,<d)20&d*676
(t2) :|
(t2) >a*18&(a,b)6 b*18&(b,<g)6 g*18&(g,a)6 a*18&(a,>g)6
(t2)  g*18&(g,a)6 a*18&(a,b)6  b (b,<g)20&g*676
(t2)
(t2) @72 v15 o4 @s4 @m30
(t2) (f,a)12&a*60 (a,d)12&d*300 r2 (d,a)5&a*91 (a,g)5&g*91 <c2
(t2) (c,>e)2&e*22 fe (e,c)4&c*500 r4 (c,>g)4&g*20a (a<,c)4&c*20d (d,e)4&e*20 g
(t2) (f,a)12&a*60 (a,d)12&d*300 r2 d2 (d,e)5&e*91 a2
(t2) (a,g)4&g*380 (g,<c)4&c*380
(t2)
(t2) @73 @v123 o5 @s6 @h96 @m53 @k1
(t2) (f,g)12&g*60 (g,c)4&c*68 (c,g)4&g*44&g1 |:2
(t2) >r2a-<ce- (e-,g)12&g*84 (g,c)4&c*68 (c,g)4&g*44
(t2) f4.>(g-,b-)12&b-*300 r1r1
(t2) <e-4. (e-,>a-)4&a-*20&a-2&a-1 r1r4. eg-a-(a-,<d-)4&d-*44 |
(t2) >b-4.(b-,<e-)4&e-*20&e-2 (e-,a-)4&a-*68e-4.(e-,>b-)4&b-*44
(t2) <(f,a)12&a*36&a2 (a,g)2&g*22a (a,f)4&f*20&f2..
(t2) (f,g)4&g*68 (g,c)4&c*20&c2&c1 :|
(t2) (d-,>b-)4&b-*68 (b-,<e-)4&e-*68 (e-,a-)4&a-*68
(t2) (a-,e-)4&e-*68 a-4 (a-,<e-)4&e-*44 (e-,>a-)4&a-*188 (a-,f)4&f*188 >
(t2) (a-,b-)24&b-*360<(a-,b-)5&b-*955 q8(b-,a-)2&a-*46 q7a- q8(a-,b-)2&b-*22
(t2)
(t2) @74 v0 o4 @s7 @h20 @m25 r2 @k0 |: v8
(t2) b-4.f*600  q7 e-4d4 q8 c4.g*696
(t2) a-4.d-*600>q7 b4b-4 q8 a-4.<e-*696 :|
(t2) @71 @m6 @h40 @s390 r1r1 v14
/
/ F M   M e l o d y
/
(t3) @71 q8 @k-2 l8 p3 v14 o4 @m6 @h40 @s390 r2.
(t3) [do]
(t3) |:2
(t3)  a*18&(a,b)6 b*18&(b,<c)6 c*18&(c,g)6 (e,g)20&g*292
(t3)  (g,e)5&e*19&e4 (e,>a)5&a*19&a1 |
(t3)  a*18&(a,<c)6 c*18&(c,g)6 g*18&(g,a)6 a*18&(a,>g)6
(t3)  g*18&(g,a)6 a*18&(a,b)6  b (b,<d)20&d*676
(t3) :|
(t3) >a*18&(a,b)6 b*18&(b,<g)6 g*18&(g,a)6 a*18&(a,>g)6
(t3)  g*18&(g,a)6 a*18&(a,b)6  b (b,<g)20&g*676
(t3)
(t3) @72 v15 o4 @s4 @m30 @k-1
(t3) (f,a)12&a*60 (a,d)12&d*300 r2 (d,a)5&a*91 (a,g)5&g*91 <c2
(t3) (c,>e)2&e*22 fe (e,c)4&c*500 r4 (c,>g)4&g*20a (a<,c)4&c*20d (d,e)4&e*20 g
(t3) (f,a)12&a*60 (a,d)12&d*300 r2 d2 (d,e)5&e*91 a2
(t3) (a,g)4&g*380 (g,<c)4&c*380
(t3)
(t3) @73 @v123 o5 @s6 @h96 @m53 @k-2
(t3) g4.c4.g4&g1 |:2r2>a-<ce-g2c4.g4 f4.>b-&b-2&b-1 r1r1
(t3) <e-4.>a-&a-2&a-1 r1r4.eg-a-<d-4> | b-4.<e-&e-2 a-4.e-4.>b-4< a2.ga f1
(t3) g4.c&c2&c1 :|
(t3) >b-4.<e-4.a-4.e-4.a-4<e-4>a-1f1>
(t3) (a-,b-)24&b-*360<(a-,b-)5&b-*955 q8(b-,a-)2&a-*46 q7a- q8(a-,b-)2&b-*22
(t3)
(t3) @74 v0 o4 @s7 @h20 @m25 r2 |: v8
(t3) b-4.f*600 q7 e-4d4   q8 c4.g*696
(t3) a-4.d-*600> q7 b4b-4 q8  a-4.<e-*696 :|
(t3) @71 @m6 @h40 @s390 r1r1 v14 @k-2
/
/ F M   M e l o d y
/
(t4) @71 q8 @k1 l8 p3 v14 o4 @m6 @h40 @s390 r16 r2.
(t4) [do]
(t4) |:2
(t4)  a*18&(a,b)6 b*18&(b,<c)6 c*18&(c,g)6 (e,g)20&g*292
(t4)  (g,e)5&e*19&e4 (e,>a)5&a*19&a1 |
(t4)  a*18&(a,<c)6 c*18&(c,g)6 g*18&(g,a)6 a*18&(a,>g)6
(t4)  g*18&(g,a)6 a*18&(a,b)6  b (b,<d)20&d*676
(t4) :|
(t4) >a*18&(a,b)6 b*18&(b,<g)6 g*18&(g,a)6 a*18&(a,>g)6
(t4)  g*18&(g,a)6 a*18&(a,b)6  b (b,<g)20&g*676
(t4)
(t4) @72 v14 o4 @s4 @m30 @k2
(t4) (f,a)12&a*60 (a,d)12&d*300 r2 (d,a)5&a*91 (a,g)5&g*91 <c2
(t4) (c,>e)2&e*22 fe (e,c)4&c*500 r4 (c,>g)4&g*20a (a<,c)4&c*20d (d,e)4&e*20 g
(t4) (f,a)12&a*60 (a,d)12&d*300 r2 d2 (d,e)5&e*91 a2
(t4) (a,g)4&g*380 (g,<c)4&c*380
(t4)
(t4) @73 v14 o5 @s6 @h96 @m53 @k-1
(t4) (f,g)12&g*60 (g,c)4&c*68 (c,g)4&g*44&g1 |:2
(t4) >r2a-<ce- (e-,g)12&g*84 (g,c)4&c*68 (c,g)4&g*44
(t4) f4.>(g-,b-)12&b-*300  r1r1
(t4) <e-4. (e-,>a-)4&a-*20&a-2&a-1 r1r4. eg-a-(a-,<d-)4&d-*44 |
(t4) >b-4.(b-,<e-)4&e-*20&e-2 (e-,a-)4&a-*68e-4.(e-,>b-)4&b-*44
(t4) <(f,a)12&a*36&a2 (a,g)2&g*22a (a,f)4&f*20&f2..
(t4) (f,g)4&g*68 (g,c)4&c*20&c2&c1 :|
(t4) (d-,>b-)4&b-*68 (b-,<e-)4&e-*68 (e-,a-)4&a-*68
(t4) (a-,e-)4&e-*68 a-4 (a-,<e-)4&e-*44 (e-,>a-)4&a-*188 (a-,f)4&f*188 >
(t4) (a-,b-)24&b-*360<(a-,b-)5&b-*955 q8(b-,a-)2&a-*46 q7a- q8(a-,b-)2&b-*22
(t4)
(t4) @74 v0 o4 @s7 @h20 @m25 r2 |: v7
(t4) b-4.f*600 q7 e-4d4 q8 c4.g*696
(t4) a-4.d-*600>q7b4b-4 q8 a-4.<e-*696 :|
(t4) @71 @m6 @h40 @s390 r1r1 v13 @k1
/
/ F M
/
(t5) @75 q8 @k-2 l8 p3 v14 o3 @m6 @h40 @s390 r2.
(t5) [do]
(t5) |:2
(t5)  ab<c g&g2&g1 e4.>a&a1 |
(t5)  a<cga> gab (b,<d)20&d*676 :|
(t5)  >ab<ga >gab (b,<g)20&g*676
(t5)
(t5) @72 v14 o4 @s4 @m30 @k0
(t5) (d,f)12&f*60 (f,>b-)12&b-*300 r2 (b-,<f)5&f*91 (f,e)5&e*91  a2
(t5) (a,c)2&c*22 dc (c,>a)4&a*500 r4 (a,e)4&e*20f (f,g)4&g*20b(b,<c)4&c*20f
(t5) (d,f)12&f*60 (f,>b-)12&b-*300 r2 b-2 (b-,<c)5&c*91 f2
(t5) (f,e)4&e*380 (e,g)4&g*380
(t5)
(t5) @76 v16 o5 @s5 @h35 @m48 @k0 q8
(t5) g4.c4.g4&g1 |:2r2>a-<ce-g2c4.g4 f4.>b-&b-2&b-1 r1r1

(t5) <e-4.>a-&a-2&a-1 r1r4.eg-a-<d-4> | b-4.<e-&e-2 a-4.e-4.>b-4< a2.ga f1
```

▶最近，「激めん」「焼そばバゴーン」「SBスナック鈴木くん・佐藤くん・田中さん」「プクポン」などを見かけなくなってしまったが，元気でやっているのだろうか。元気ならまた会いたいものだ。

浅場　誠志(20)神奈川県

```
(t5) g4.c&c2&c1 :|
(t5) >b-4.<e-4.a-4.e-4.a-4<e-4>a-1f1
(t5)
(t5)  @77 v12 @k0 @m1 o4
(t5) |:8 q8e-4 q7e- q8f | rfrf :|
(t5)  @78 l8 q3 r2 o4
(t5) |:2
(t5) |:4 v14p1b-r v13p3b-v14p1b- p3v7b-p1v14b- p3v7b-p1v
14b-:|
(t5) |:4 v14p1a-r v13p3a-v14p1a- p3v7a-p1v14a- p3v7a-p1v
14a-:|
(t5) |:4 v14p1g-r v13p3g-v14p1g- p3v7g-p1v14g- p3v7g-p1v
14g-:|
(t5) |:4 v14p1e r v13p3e v14p1e  p3v7e p1v14e  p3v7e p1v
14e :|
(t5) :| q7 p3 v10 @k0
(t5) >f(f,b)4&b*20 (b,<e-)4&e-*20 (e-,>a-)4&a-*20 (a-,<d
-)4&d-*20 (d-,g-)4&g-*20
(t5) >b- (b-,<e-)4&e-*20 >
(t5) f (f,<c)4&c*20 (c,f)4&f*20 (f,>a)4&a*20 (a,<d)4&d*2
0 (d,g)4&g*20
(t5) >b (b,<e)4&e*20
(t5) @75 @k-2 v10 @m6 @s390 @h40 q8
/
/ F M
/
(t6) @75 q8 @k-3 l8 p3 v13 o3 @m6 @h40 @s390 r2.
(t6) [do]
(t6) |:2
(t6)  ab<c g&g2&g1 e4.>a&a1 |
(t6)  a<cga> gab (b,<d)20&d*676 :|
(t6)  >ab<ga >gab (b,<g)20&g*676
(t6)
(t6) @72 v13 o4 @s4 @m30 @k1 r16
(t6) (d,f)12&f*60 (f,>b-)12&b-*300 r2 (b-,<f)5&f*91 (f,e
)5&e*91  a2
(t6) (a,c)2&c*22 dc (c,>a)4&a*500 r4 (a,e)4&e*20f (f,g)4
&g*20b(b,<c)4&c*20f
(t6) (d,f)12&f*60 (f,>b-)12&b-*300 r2 b-2 (b-,<c)5&c*91
f2
(t6) (f,e)4&e*380 (e,g)4&g*380
(t6)
(t6) @76 v15 o5 @s5 @h35 @m48 @k2 q8
(t6) g4.c4.g4&g1 |:2r2>a-<ce-g2c4.g4 f4.>b-&b-2&b-1 r1r1

(t6) <e-4.>a-&a-2&a-1 r1r4.eg-a-<d-4> | b-4.<e-&e-2 a-4.
e-4.>b-4< a2.ga f1
(t6) g4.c&c2&c1 :|
(t6) >b-4.<e-4.a-4.e-4.a-4<e-4>a-1f2...
(t6)
(t6)  @77 v12 @k0 @m1 o4
(t6) |:8 q8c4 q7c q8d | rdrd :|
(t6)  @78 l8 q3 r2 o4
(t6) |:2
(t6) |:4 v14p2f r v13p3f v14p2f  p3v7f p2v14f  p3v7f p2v
14f :|
(t6) |:4 v14p2e-r v13p3e-v14p2e- p3v7e-p2v14e- p3v7e-p2v
14e-:|
(t6) |:4 v14p2d-r v13p3d-v14p2d- p3v7d-p2v14d- p3v7d-p2v
14d-:| >
(t6) |:4 v14p2b r v13p3b v14p2b  p3v7b p2v14b  p3v7b p2v
14b :| <
(t6) :| q7 p3 @k-1 v11
(t6) >f(f,b)4&b*20 (b,<e-)4&e-*20 (e-,>a-)4&a-*20 (a-,<d
-)4&d-*20 (d-,g-)4&g-*20
(t6) >b- (b-,<e-)4&e-*20 >
(t6) f (f,<c)4&c*20 (c,f)4&f*20 (f,>a)4&a*20 (a,<d)4&d*2
0 (d,g)4&g*20
(t6) >b (b,<e)4&e*20
(t6) @75 @k-2 v10 @m6 @s390 @h40 q8
/
/ F M
/
(t7) @79 q8 @k-3 l8 p3 v16 o3 @m50 @s1 r2.
(t7) [do]  l4
(t7) v10c8&v11c8& v13c8&v14c8& v15c2&
(t7) v14c&v13c&v12c&v12c& c&v11c&v10c&v9c&  v8c&v7c&v6c&
v5c& v4c&v3c&v2c&v1c
(t7) @80 o4  v10 q7 @m1 @k1 |:3r1:| l8
(t7) a*188&(a,d)4 d*188&(d,g)4 g*188&(g,c)4 c1>b1&b1 r1r
1
(t7)
(t7) @78 v13 q3 o4 @k1
(t7) |:8 v13p1av12p2av13p3ap1ar2  p2av12p1av13p3ap2ar2 :
|
(t7)
(t7) @76 v16 o5 @s5 @h35 @m48 @k0 q8
(t7) e-4.>a-4.<e-4&e-1 |:2
(t7) >r2a-<ce- e-2>a-4.<e-4 d-4.>g-&g-2&g-1 r1r1
(t7) b4.e&e2&e1 r1r4. eg-a-b-4 |
(t7) f4.b-&b-2<e-4.>b-4.f4 <c2.gac1
(t7) e-4.>a-&a-2&a-1    :|
(t7) f4.b-4.<e-4.>b-4.<e-4b-4 f1c1
(t7)
(t7)  @77 v12 @k-1 @m1 o3
(t7) |:8 q8a-4q7a-q8b-| rb-rb- :|
(t7)
(t7) @74 v0 o4 @s6 @h20 @m25 r2 @k1 r8 |: v7
(t7) b-4.f*600  q7 e-4d4 q8 c4.g*696
(t7) a-4.d-*600>q7 b4b-4 q8 a-4.| <e-*696 :| e-*672
(t7) @81 v16 @k1 p3 o2 q7
(t7) >f(f,b)4&b*20 (b,<e-)4&e-*20 (e-,>a-)4&a-*20 (a-,<d
-)4&d-*20 (d-,g-)4&g-*20
(t7) >b- (b-,<e-)4&e-*20 >
(t7) f (f,<c)4&c*20 (c,f)4&f*20 (f,>a)4&a*20 (a,<d)4&d*2
0 (d,g)4&g*20
(t7) >b (b,<e)4&e*20
(t7) @79 q8 @k-3 l8 p3 v16 o3 @m40 @s2
/
/ F M
/
(t8) @79 q8 @k0 l8 p3 v16 o2 @m50 @s1 r2.
(t8) [do] l4
(t8) v10b8&v11b8& v13b8&v14b8& v15b2&
(t8) v14b&v13b&v12b&v12b& b&v11b&v10b&v9b&  v8b&v7b&v6b&
v5b& v4b&v3b&v2b&v1b
(t8) @80 o4 v10 q8 @m1 @k-2 |:3r1:| l8 q7
(t8) a1d1g1c1>b1&b1 r1 @81 v16 @k0 o1r2 q6f<q8f>q6g<q8g>

(t8) @78 v13 q3 o4 @k-1
(t8) |:8 v13p1av12p2av13p3ap1ar2  p2av12p1av13p3ap2ar2 :

(t8) @76 v15 o5 @s5 @h35 @m48 @k2 r16 q8
(t8) e-4.>a-4.<e-4&e-1 |:2
(t8) >r2a-<ce- e-2>a-4.<e-4 d-4.>g-&g-2&g-1 r1r1
(t8) b4.e&e2&e1 r1r4. eg-a-b-4 |
(t8) f4.b-&b-2<e-4.>b-4.f4 <c2.gac1
(t8) e-4.>a-&a-2&a-1  :|
(t8) f4.b-4.<e-4.>b-4.<e-4b-4 f1c2...
(t8)
(t8)  @77 v12 @k2 @m1 o4
(t8) |:8 q8e-4 q7e- q8f | rfrf :|
(t8)
(t8) @74 v0 o4 @s6 @h20 @m25 r2 @k3 r8. |: v6
(t8) b-4.f*600  q7 e-4d4 q8 c4.g*696
(t8) a-4.d-*600>q7 b4b-4 q8 a-4.| <e-*696 :| e-*660
(t8)
(t8) @78 q7 p3 @k1 v9 r16 @m1
(t8) >f(f,b)4&b*20 (b,<e-)4&e-*20 (e-,>a-)4&a-*20 (a-,<d
-)4&d-*20 (d-,g-)4&g-*20
(t8) >b- (b-,<e-)4&e-*20 >
(t8) f (f,<c)4&c*20 (c,f)4&f*20 (f,>a)4&a*20 (a,<d)4&d*2
0 (d,g)4&g*20
(t8) >b (b,<e)4&e*8
(t8) @79 q8 @k0 l8 p3 v16 o2 @m50 @s1
/
/ A D   P C M*************************Bass Drum
/
(t9) l8p3o0 r2.
(t9) [do]
(t9) |:|:3 ccr2cc:| ccrc | rrcr:| rrrc
(t9) |:|:rcr2rc|rcr2.:||rcrc rcr4:| rcr4 c4r4
(t9) |:4|:4c4rc|r4.c:| r2:|
(t9) |:32 ccrc | rcr4:| r2
(t9) |:8 ccrc|rcr4 :| r2
(t9) |:16 c4rc rcrc | cc4c rcrc:| c4rcr2
(t9) |:2c4rc r2:|
/
/ A D   P C M*************************Snare Drum
/
(t10) l8p3o0 r2.
(t10) [do]
(t10) |:2 r1r1r1 r2.d4:| |:13r4d4:| r4dd4rd4 rd4r
(t10) |:15 r4d4 :| r2
(t10) |:15 r4d4 :| r4d16d16d16d16
(t10) |:30 r4d4 :| r4d4 r4d16d8.
(t10) |:30 r4d4 :| r4d4 f16f16e16e16 e16d16d16d16
(t10) |:15 r4d4 :| r4d16d8.
(t10) |:31 r4d4 :| r4d16d16d16d16
(t10) |:31 r4d4 :| d16d16d16d16d16d8.
(t10)  r2.d4 r1
/
/ A D   P C M*************************Tom Tom
/
(t11) l8p3o0 r2.
(t11) [do]
(t11) |:15r1:| r2.<d4>
(t11) |:|:3r1:| r2gab4 |:3r1:||r2gab<c> :| r1
(t11) |:|:3r1:| r2.b<c> |
(t11) |:3r1:| r2.<c4>   :|
(t11) |:r1:| rb4r r4<c4> rb4r r4.<f16c16>
(t11) |:|:3r1:| r2.b<c> |
(t11) |:3r1:| r2.<c4>   :|
(t11) |:r1:| rb4r r4<c4> rb4r r2
(t11) |:7r1:|  r2..g16b16
(t11) |:2|:3|:3r1:| r2<f4rf>:|||:4r1:|:| |:3r1:| r2..<f1
6g16>
(t11) |:2 b4.<c4>r4.:|
/
/ M I D I   R h y t h m**************Bass Drum
/
(t12) l8p3o2 @u120 @v127 @d0 r2.
(t12) [do]
(t12) |:|:3 ccr2cc:| ccrc | rrcr:| rrrc
(t12) |:|:rcr2rc|rcr2.:||rcrc rcr4:| rcr4 c4r4
(t12) |:4|:4c4rc|r4.c:| r2:|
(t12) |:32 ccrc | rcr4:| r2
(t12) |:8 ccrc|rcr4 :| r2
(t12) |:16 c4rc rcrc | cc4c rcrc:| c4rcr2
(t12) |:2c4rc r2:|
/
/ M I D I   R h y t h m**************Snare Drum
/
(t13) l8p3o2 @u120 @d0 r2.
(t13) [do]
(t13) |:2 r1r1r1 r2.d4:| |:13r4d4:| r4dd4rd4 rd4r
(t13) |:15 r4d4 :| r2
(t13) |:15 r4d4 :| r4d16d16d16d16
(t13) |:30 r4d4 :| r4d4 r4d16d8.
(t13) |:30 r4d4 :| r4d4 |:8d16:|
(t13) |:15 r4d4 :| r4d16d8.
(t13) |:31 r4d4 :| r4d16d16d16d16
(t13) |:31 r4d4 :| d16d16d16d16d16d8.
(t13)  r2.d4 r1
/
/ M I D I   R h y t h m**************Hihat
/
(t14) l8p3o2 @u120 @d0 r2.q8
(t14) [do]
(t14) |:4eg:| fg|:7eg:| fgeg eg'fa'e
(t14) |:4eg:||:2fg|:3eg:|:| fgeg eg'fa'e
(t14) |:7fgeg eg|eg:| e'fa'|:4eg:|
(t14)
(t14) |:16 ggea eg| eg :| r4
(t14)
(t14) |:39 eegf eaeg:| eegfr2
(t14)
(t14) |:2|:7eege egee  efge egee :||:2 eege egee:|:|
(t14) f4gf4gf4 f4gf4gr4
/
```

▶私の周囲の連中は，どいつもこいつもX68000をいっぱい持っている。4台を使っている私でさえ所有者TOPではない。　下田　達也(25)三重県

```
/ M I D I   R h y t h m**************Cymbal
/
(t15) l8p3o3 @u120 @d0 r2.
(t15) [do]
(t15) 'c4d'r2.|:7r1:| c4r2.|:7r1:|
(t15) |:2d4r2.|:7r1:|:|
(t15) |:2|:4c4r2.||:3r1:|:| r1d4r2.c4r2.:|
(t15) 'c4d'r2. |:7r1:|
(t15) |:4c4r2. |:7r1:|:|
(t15)  r1r2.@u100'c4d'@u120
/
/ P C M*****************************************CH11
/
(t16) q8 @23 l8 v12 o1 r2. @u101 @p63
(t16) [do]
(t16) |:15r1:|
(t16)  r2 @u70 >q6f<q8f>q6g<q8g @u101
(t16) q7 |:2|:4b-rrb-r2:||:4arrar2:|:|
(t16) |:2|:2 a-a-ra- ra-a-a-  a-a-ra- r<a->a-a- :|
(t16)    |:2 g-g-rg- rg-g-g-  g-g-rg- r<g->g-g- :|
(t16)    |:2 e e re  re e e   e e re  r<e >e e  :|
(t16)    |:2 f f rf  rf f f | f f rf  r<f >f f  :| ffrq1
f q7r<f>ff
(t16) :|
(t16)    |:8 q8b-4q7b-q8b-| rb-rb-:| r2
(t16) |:2|:4 q8b-4q7b-q8b-  rb-rb-:|
(t16)    |:4 q8a-4q7a-q8a-  ra-ra-:|
(t16)    |:4 q8g-4q7g-q8g-  rg-rg-:|
(t16)    |:4 q8e4 q7e q8e   re re :|
(t16) :| q7
(t16) fb<e->a- <d-g->b-<e->
(t16) f<cf>a   <dg>b<e>
/
/ L A*********************************************CH02
/
(t17) q8 @017 l8 v12 @u92 l8 o4 @p64 r2.
(t17) [do] q6
(t17) |:2
(t17) |:4@p58b<@p62c@p66g>@p70b <@p58c@p62g>@p66b<@p70c
(t17) @p58g>@p62b<@p66c@p70g >@p58b<@p62c@p66g|1@p70<c>:
||2@p70a:||3@p70b:||4@p70e:|
(t17) :|
(t17) @002 q7 @u117
(t17) |:2
(t17) |:2r2r@p60d@p62e@p64d | r2r@p68<c>@p66f@p64g:| r2r
@p68e@p66f@p64g
(t17) |:2r2r@p60d@p62e@p64d | r2r@p68d@p66e@p64d:|   r2r
@p68<c>@p66g@p64e
(t17) :|
(t17) |:73 r1:|
(t17) q8 @017 l8 v12 @u90 l8 o4 @p64 r1
/
/ L A*********************************************CH03
/
(t18) q8 @98 l8 v13 @u105 l8 o5 @p64 r2. @m1
(t18) [do]
(t18) |:8r1:|@p53 q7 a1d1g1c1>b1&b1 r1 @71 @u92 @p62 q7
r1 @m80
(t18) o4|:|:3arrar2:| <crrcr2> grrgr2 brrbr2 errer2 grrg
r2:|
(t18) o5|:|:4f4.e-re-r4:| |:4e-4.d-rd-r4:| |:4d-4.>brbr4
<:|
(t18) |:2>b-4.b-rb-r4:| a4.arar4 a4.a&a2:|
(t18) o5  |:8q8e-4q7e-q8f| rfrf:| r2
(t18) |:o5|:4q8e-4q7e-q8f  rfrf:|
(t18)      |:4q8e-4q7e-q8g  rgrg:|
(t18)      |:4q8d-4q7d-q8e- re-re-:|
(t18)      |:4>q8b4q7bq8<d- rd-rd-:|:|
(t18) r1 q8 @98 l8 v13 @u105 l8 o5 @p64 r1 @m1
/
/ L A*********************************************CH04
/
(t19) q7 @71 l8 v13 @u92 l8 o3 @p64 r2. @m80
(t19) [do]
(t19) |:16r1:|
(t19) o4|:|:3frrfr2:| grrgr2 errer2 grrgr2 crrcr2 drrdr2
:|
(t19) o5|:|:4d-4.crcr4:| |:4c4.>b-rb-r4<:| |:4>b-4.a-ra-
r4<:|
(t19) |:2>f4.frfr4<:|>f4.frfr4 f4.f&f2:|
(t19) o5  |:8q8c4q7cq8d| rdrd:| r2
(t19) |:o5|:4q8c4q7cq8d  rdrd:|
(t19)      |:4q8c4q7cq8e- re-re-:|
(t19)      |:4>q8b-4q7b-q8<c rcrc:|
(t19)      |:4>q8a-4q7a-q8b- rb-rb-<:|:|
(t19) r1r1
/
/ L A*********************************************CH05
/
(t20) q7 @71 l8 v13 @u92 l8 o3 @p66 r2. @m80
(t20) [do]
(t20) |:16r1:|
(t20) o4|:|:3drrdr2:| errer2 crrcr2 drrdr2>grrgr2 brrbr2
<:|
(t20) o5|:|:4b-4.a-ra-r4:| |:4a-4.g-rg-r4:| |:4g-4.erer4
:|
(t20) |:2e-4.e-re-r4:| c4.crcr4 c4.rr2:|
(t20) o4  |:8q8a-4q7a-q8b-| rb-rb-:| r2
(t20) |:  |:4q8a-4q7a-q8b-  rb-rb-:|
(t20)     |:4q8a-4q7a-q8<c  rcrc> :|
(t20)     |:4q8g-4q7g-q8a-  ra-ra-:|
(t20)     |:4q8e4q7e q8g- rg-rg- :|:|
(t20) r1r1
/
/ L A*********************************************CH06
/
(t21) q8 @29 l8 v14 @u105 l8 o2 @p63 r2. @m110
(t21) [do]
(t21) b*1536 |:4r1:| v16 @102  @m40 @u125 o6 |:4r1:| q8
(t21) r1 @p20a4.d4.<c4> |:6r1:|
(t21) r1     a4.d4.<c4> |:5r1:| r2<cdeg
(t21) @74 @m50 @u66 q7 @p63 l8 o6 v15
(t21) |:|:4r4.crcr4:||:4>r4.b-rb-r4<:||:4>r4.a-ra-r4<:|
(t21) |:2>r4.b-rb-r4<:|> r4.frfr4 r4.a&a2:| r1
(t21) q8 @29 l8 v13 @u105 l8 o2 @p63 @m110
(t21) |:41r1:|
/
/ L A*********************************************CH07
/
(t22) q8 @29 l8 v14 @u105 l8 o3 @p65 r2. @m110
(t22) [do]
(t22) c*1536 |:4r1:| v16 @102 @m40 @u120 o6 |:4r1:| q8
(t22) r1r  @p107a4.d4 <c4> |:6r1:|
(t22) r1r        a4.d4 <c4> |:5r1:| r2r<cde
(t22) @74 @m50 @u66 q7 @p63 l8 o5 v15
(t22) |:|:4r4.a-ra-r4:||:4r4.g-rg-r4:||:4r4.erer4:|
(t22) |:2r4.frfr4:|r4.crcr4 r4.f&f2:| r1
(t22) q8 @29 l8 v13 @u105 l8 o3 @p65 @m110
(t22) |:41r1:|
/
/ P C M*******************************************CH12
/
(t23) q8 @35 l8 v7 @u120 l8 o5 @p64 r2.@q7
(t23) [do] q7
(t23) |:16r1:|
(t23) o2|:|:3arrar2:| <crrcr2> grrgr2 brrbr2 errer2 grrg
r2:|
(t23) o3|:|:4f4.e-re-r4:| |:4e-4.d-rd-r4:| |:4d-4.>brbr4
<:|
(t23) |:2>b-4.b-rb-r4:| a4.arar4 a4.a&a2:| @u122
(t23) o3  |:8  e-4q7e-q8f| rfrf:| r2
(t23) |:o3|:4  e-4q7e-q8f  rfrf:|
(t23)      |:4  e-4q7e-q8g  rgrg:|
(t23)      |:4  d-4q7d-q8e- re-re-:|
(t23)      |:4>  b4q7bq8<d- rd-rd-:|:| @u120
(t23) r1r1
/
/ P C M*******************************************CH12
/
(t24) q8 @35 l8 @u120 l8 o5 r2.@q7
(t24) [do] q7
(t24) |:16r1:|
(t24) o2|:|:3frrfr2:| grrgr2 errer2 grrgr2 crrcr2 drrdr2
:|
(t24) o3|:|:4d-4.crcr4:| |:4c4.>b-rb-r4<:| |:4>b-4.a-ra-
r4<:|
(t24) |:2>f4.frfr4<:|>f4.frfr4 f4.f&f2:| @u122
(t24) o3  |:8  c4q7cq8d| rdrd:| r2
(t24) |:o3|:4  c4q7cq8d  rdrd:|
(t24)      |:4  c4q7cq8e- re-re-:|
(t24)      |:4>  b-4q7b-q8<c rcrc:|
(t24)      |:4>  a-4q7a-q8b- rb-rb-<:|:| @u120
(t24) r1r1
/
/ P C M*******************************************CH12
/
(t25) q8 @35 l8 @u120 l8 o5 r2.@q7
(t25) [do] q7
(t25) |:16r1:|
(t25) o2|:|:3drrdr2:| errer2 crrcr2 drrdr2>grrgr2 brrbr2
<:|
(t25) o3|:|:4b-4.a-ra-r4:| |:4a-4.g-rg-r4:| |:4g-4.erer4
:|
(t25) |:2e-4.e-re-r4:| c4.crcr4 c4.rr2:| @u122
(t25) o2  |:8  a-4q7a-q8b-| rb-rb-:| r2
(t25) |:  |:4  a-4q7a-q8b-  rb-rb-:|
(t25)     |:4  a-4q7a-q8<c  rcrc >:|
(t25)     |:4  g-4q7g-q8a-  ra-ra-:|
(t25)     |:4  e4q7eq8g- rg-rg- :|:| @u120
(t25) r1r1
/
/ P C M*******************************************CH13
/
(t26) q8 @035 l8 v8 l8 o4 @p64 r2. @m75
(t26) [do] q6 @u104
(t26) |:2
(t26) |:4@p58b<@p62c@p66g>@p70b <@p58c@p62g>@p66b<@p70c
(t26) @p58g>@p62b<@p66c@p70g >@p58b<@p62c@p66g|1@p70<c>:
||2@p70a:||3@p70b:||4@p70e:|
(t26) :|   q6 @u108
(t26) |:2
(t26) |:2r2r@p60d@p62e@p64d | r2r@p68<c>@p66f@p64g:| r2r
@p68e@p66f@p64g
(t26) |:2r2r@p60d@p62e@p64d | r2r@p68d@p66e@p64d:|   r2r
@p68<c>@p66g@p64e
(t26) :|
(t26) v7 @u105 o5 q7 @p63
(t26) |:
(t26) |:4r4.'a-<c'r'a-<c'r4:|
(t26) |:4r4.'g-b-'r'g-b-'r4:|
(t26) |:4r4.'ea-'r'ea-'r4:|
(t26) |:2r4.'fb-'r'fb-'r4:|
(t26)    r4.'cf'r'cf'r4 r4.'f*120a'
(t26) :|
(t26) |:4r1:|r2 v12 r2|:3r1:|
(t26) o4 @u59 @p62  |:
(t26) q8 b-4.f*600   q7e-4q8d4  c4.g*696
(t26) q8 a-4.d-*600> q7b4q8b-4  a-4.<e-*696 :|
(t26)    @u54 o4 @p64 q7
(t26) fb<e->a- <d-g->b-<e->
(t26) f<cf>a   <dg>b<e>
(t26) q8 @035 l8 v8 l8 o4 @p64 @m75
/
/ P C M*******************************************CH14
/
(t27) q8 @028 l8 v10 l8 o1 @p64 r2. @m3
(t27) [do] q8 @u105
(t27) |:2a1&a1 f1&f1|g1&g1&g1&g1:| g1&g1&g1&g2 @u125 q6f
<q8f>q6g<q8g
(t27) @31 o5 @u93 l32 |:12r1:|
(t27) r4
(t27) |:7@p64c&@p50c&@p46c&@p32c& @p46c&@p50c&@p64c&@p78
c& @p82c&@p96c&@p82c&@p78c&:|
(t27)    @p64c&@p50c&@p46c&@p32c  r1
(t27)
(t27) |:32r1:|
(t27) |:4r1:|r2 @34 v12 @m30 r2|:3r1:|
(t27) o5 @u59 @p66  |:
(t27) q8 b-4.f*600   q7e-4q8d4  c4.g*696
(t27) q8 a-4.d-*600> q7b4q8b-4  a-4.<e-*696 :|
(t27) r1
(t27) q8 @028 l8 v10 l8 o1 @p64 @m3 r1
```

▶いやあ，データレコーダを駆使するMZ-2500用のプログラム，やってくれましたね。MZ-2500は持っていないけどすごいぞ。　阪長　俊之(25)京都府

```
/
/ P C M***************************************CH15
/
(t28) q8 @031 l8 v0 l8 o5 @p64 r2. @m45
(t28) [do] l4 @p66 @u100
(t28) |:
(t28) v5g1& v6g2&v7g2& v8g2&v9g2& v10g1&
(t28) g1     v9g1&        v8g2&v7g2& v6g2&v5g2
(t28) :|
(t28) v7 o3 l8 @u107
(t28) @p60'b-4<df'r @p62'd*312fb-' r2@p68'b-2<df'@p66'g2
<ce'@p62'<c2eg'
(t28) @p60'c*576ea'r1
(t28) @p60'b-4<df'r @p62'd*312fb-' r2@p68'd2fb-'@p66'f2b
-<d'@p62'b-2<df'
(t28) @p60'g*576<ce'r1
(t28) |:72 r1:|
(t28)  @p64 @u105 o5 r1r1

(t1) [loop]
(t2) [loop]
(t3) [loop]
(t4) [loop]
(t5) [loop]
(t6) [loop]
(t7) [loop]
(t8) [loop]
(t9) [loop]
(t10) [loop]
(t11) [loop]
(t12) [loop]
(t13) [loop]
(t14) [loop]
(t15) [loop]
(t16) [loop]
(t17) [loop]
(t18) [loop]
(t19) [loop]
(t20) [loop]
(t21) [loop]
(t22) [loop]
(t23) [loop]
(t24) [loop]
(t25) [loop]
(t26) [loop]
(t27) [loop]
(t28) [loop]

(p)
```

リスト2　ドラゴンセイバー用コンフィグファイル

```
.00c =dsb.pcm,v100
.00d =dsnar.pcm,v172
.00e =dsnar.pcm,v135
.00f =dsnar.pcm,v100
.00g =atkt4.pcm,p4,v142
.00a =atkt3.pcm,p2,v142
.00b =atkt2.pcm,p1,v142
.01c =atkt1.pcm,p-3,v142
.01d =atkt4.pcm,p4,v107
.01e =atkt3.pcm,p2,v107
.01f =atkt2.pcm,p1,v107
.01g =atkt1.pcm,p-3,v107
```

リスト3　ドラゴンセイバー用カウンタ表示

```
 1:000000C0 00004F80   2:000000C0 00004F80   3:000000C0 00004F80   4:000000CC 00004F80
 5:000000C0 00004F80   6:000000C0 00004F80   7:000000C0 00004F80   8:000000C0 00004F80
 9:000000C0 00004F80  10:000000C0 00004F80  11:000000C0 00004F80  12:000000C0 00004F80
13:000000C0 00004F80  14:000000C0 00004F80  15:000000C0 00004F80  16:000000C0 00004F80
17:000000C0 00004F80  18:000000C0 00004F80  19:000000C0 00004F80  20:000000C0 00004F80
21:000000C0 00004F80  22:000000C0 00004F80  23:000000C0 00004F80  24:000000C0 00004F80
25:000000C0 00004F80  26:000000C0 00004F80  27:000000C0 00004F80  28:000000C0 00004F80
```

リスト4　ガラガラヘビがやってくる

日本音楽著作権協会(出)許諾第9270765-201号

```
  10 dim str a(50)[256]:dim str aa(50)[256]
  20 /*=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
  30 /**-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
  40 m_init()
  50 for i=1 to 8:m_alloc(i,3000):m_assign(i,i):next
  60 /*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*
  70 /*-/-/-/-/-/-/-/ｳﾞｫｰｶﾙ/-/-/-/-/-/-/-/-/-/-/-/-/-/-
  80 a(0)="@78o3v13q818 t165  r1r1r1r1r1r1r1 r4
  90 a(1)="a+a+<c+c+>b+4a+g+a+4e+g+a+4r4a+a+<c+c+>b+b+a+g+<c+4c
+d+e+4r4
 100 a(2)="d+&c+d+c+d+4.e+4>a+a+g+a+4r4<e+4.e+d+4.d+c+>a+4a+&a+
4r4r1r1
 110 a(3)="a+a+<c+c+>b+4a+g+a+4a+g+a+4r4a+a+<c+c+>b+b+a+g+<c+c+
c+d+e+4r4
 120 a(4)="d+c+d+c+d+4r4e+>a+a+g+a+4r4<e+4.e+d+4.d+c+>a+4a+&a+4
r4r1r1
 130 a(5)="<d+d+d+d+d+c+d+e+e+e+g+e+4r4d+d+d+d+4{c+d+}4e+2.
r4d+d+d+d+d+c+d+e+e+e+g+e+4r4c+>a+<c+>a+<c+4d+4e+1r1
 140 a(6)=">|:a+4r4ra+<e+d+4r2.r>a+4r4rr<e+d+4r2.r:|>
 150 a(7)="r1r1r1r1
 160 a(8)="a+a+<c+c+>b+4a+g+a+a+e+g+a+4r4a+a+<c+c+>b+4a+g+<c+c+
c+d+e+4r4 d+&c+d+c+d+4.d+e+>a+a+g+a+4r4<e+4.e+d+4.d+c+>a+4a+&a+4
r4r1r1
 170 a(9)="a+a+<c+c+>b+4a+g+a+a+e+g+a+4r4a+a+<c+4>b+b+a+g+<c+c+
c+d+e+4r4 d+c+d+c+d+4d+d+e+>a+a+g+a+4r4<e+4.e+d+4.d+c+>a+4a+&a+4
r4r1r1
 180 a(10)="<d+d+d+d+d+d+c+d+e+4e+g+e+4r4d+d+d+d+c+4d+4e+2.r4d+
d+d+d+d+d+c+d+e+e+e+g+e+4r4c+>a+<c+>a+<c+4d+4e+1r1
 190 a(11)=a(6)+"r1r1
 200 a(12)="a+a+<c+c+>b+4a+g+a+4e+g+a+4r4a+a+<c+c+>b+4a+g+<c+4c
+d+e+4r c+d+c+d+c+d+4.e+4>a+a+g+a+4r4<e+4.e+d+4.d+c+>a+4a+&a+4r4
r1r1
 210 for i=0 to 12:m_trk(1,a(i)):next
 220 /*=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
 230 /**-*-*-*-*-*-*-*-ﾍﾞｰｽ*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 240  aa(0)="@73o2v14q714t143 r
 250  aa(1)="o2{a+&a+&a+a+}{<a+&a+>a+r}
 260 aa(2)="r8{<e+r}8{g+&e+&g+&e+}
 270 aa(3)=aa(1)+"<{e+rd+r}{c+&d+&c+&c+}
 280 aa(4)=aa(1)+aa(2)+aa(1)
 290 aa(5)="<{e+e+d+d+}{c+d+&d+&c+}
 300 aa(6)="o3e+2d+2c+2>b+2<
 310 aa(7)="r1r218e+e+{d+c+}{>a+g+}
 320 aa(8)="18o2q4 a+<c+e+c+>g+b+<d+>b+
 330 aa(9)="a+<c+e+c+>a+<c+e+>{g+a}
 340 aa(10)=aa(8)+"<c+e+g+e+c+e+g+{c+d}
 350 aa(11)="d+f+a+f+d+f+a+f+>a+<c+e+c+>a+<c+e+c+
 360 aa(12)="e+e+{e+e+}e+d+d+{d+d+}d+>a+<c+e+c+>a+<c+e+>{g+a}
 370 aa(13)="{a+a+}{a+r}a+{a+a+}{a+r}a+{a+a+}{a+r}
 380 aa(14)="r2.o4 a+4
 390 aa(15)=aa(8)+aa(9)+aa(10)
 400 aa(16)=aa(11)+aa(12)
 410 aa(17)="o2 a+<c+e+c+>a+<c+e+c+>a+<c+d+c+>a+<c+{d+c+}>a+
 420 aa(18)="g+g+g+{g+g+}g+g+g+g+&<c+c+c+{c+c+}c+c+c+c+&d+d+d+{
d+d+}d+d+d+d+&>a+a+a+a+a+a+a+{a+a+}
 430 aa(19)="g+g+g+g+g+g+g+g+&<c+c+c+{c+c+}c+c+c+c+&f+f+f+f+f+f
+f+{f+f+}
 440 aa(20)="e+e+r4e+e+r>{c+d+}e+4r2r q7{g+a}
 450 aa(21)="o2q7 a+4r2r{g+a}a+a+
 460 aa(22)="r2<e+&{d+c+}>a+a+
 470 aa(23)="r2r{g+a}a+a+r4<{rg+&}a+e+{d+c+}
 480 aa(24)=aa(21)+aa(22)
 490 aa(25)="r2r{g+a}a+a+r2r<{c+d+}
 500 aa(26)="e+4.re+4.re+4e+4e+4e+4
 510 aa(27)=aa(7)
 520 aa(28)=aa(15)+aa(16)
 530 aa(29)=aa(13)+aa(14)
 540 aa(30)=aa(28)+aa(17)
 550 aa(31)=aa(18)+aa(19)+aa(20)+aa(21)
 560 aa(32)=aa(22)+aa(23)+aa(24)
 570 aa(33)=aa(25)+aa(26)+aa(28)
 580 aa(34)=aa(17)+aa(17)
 590 aa(35)=aa(6)+"r2.o2q8a+
 600 for i=0 to 35:m_trk(2,aa(i)):next
 610 /**+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+*+
 620 /*/*/*/*/*/*/*/*ｽﾈｱ*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*
 630 aa(0)="@74q8o3v1514  {rrrrrccc}
 640 aa(1)="|:3rcrc:|
 650 aa(2)="|:3rcrc:|
 660 aa(3)="r1r2{cccc}{cc}
 670 aa(4)=aa(2)+"rc{rc}c"+aa(1)+"rcrc{cccr}{crcc}{crcr}{cccr}
 680 aa(5)="r2.c
 690 aa(6)=aa(1)+aa(2)+"rcrcrcr{cc}
 700 aa(7)="rcrcr{cc}{cccc}{c>aar}<
 710 aa(8)=aa(1)+aa(2)
 720 aa(9)="rcrc{cc}r{cc}rcr2.
 730 aa(10)="|:3v7cv15{c&cc-c&}{ccc-&c-}{cr}:|
 740 aa(11)="v7cv15{c&cc-c&}{ccc-&c-}{c&cc-c-}
 750 aa(12)="|:4v7cv15{c&cc-c&}{ccc-&c-}{cr}:|
 760 aa(13)="rcrccccc
 770 aa(14)="r1r{c-ccc&c&c}{cc}c
 780 aa(15)=aa(4)+aa(5)+aa(6)
 790 aa(16)="rcrcr{cc}{c-c-cc&c&c}{cc}
 800 aa(17)=aa(8)+"rcr{c&c&cc-c-c}{cc}{rrrc-c-c}{cc}{rrrc-c-c}c
r{rrrc-c-c}c
 810 aa(18)=aa(10)+aa(11)
 820 aa(19)=aa(12)+aa(13)
 830 aa(20)=aa(8)+"rcrcrc{rrcc}{cccc}"+aa(8)
 840 aa(21)=aa(5)
 850 for i=0 to 21:m_trk(3,aa(i)):next
 860 /**-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 870 /**-*-*-*-*-*-*-*-*-*-*-*-*ﾄﾗﾝﾍﾟｯﾄ*-*-*-*-*-*-*-*-*-*-
 880 aa(0)="@75q8o5v1514 t145 r
 890 aa(1)="o5r{a+&a+&a+&a+&a+&a+&e+&>a+}r2{a+ra+r}<{c+rc+r}{e+
&e+&e+e+}r
 900 aa(2)=aa(1)
 910 aa(3)="e+2d+2c+2>b+2
 920 aa(4)="o5r1r2{e+e+}{d+c+>a+g+}
 930 aa(5)="r1r1r1r1
 940 aa(6)="o5r2{rrc+&d+}{c+&d&d+r}r2{rr>g+&a+}{g+&a&a+r}
 950 aa(7)="r1r2{rrg+&a+}{b+&<c+&d+&e+}{a+a+a+r}{a+ra+a+}{a+ra+
r}{a+a+a+r}
 960 aa(8)="o5{e+&e+&e+&e+&e&d&c+&c+&}{c&c&>b&b&a+&b+&<c+&}{d&d
+&e&e+&f+&a&b+&}
 970 aa(9)="{a+r}
 980 aa(10)="|:r1r2o5{g+&a&a+e+}{g+&a&a+&a+}:|
 990 aa(11)="o5r2{rrd+&d+&}{d+&d+&d+&e&e+&f+&a+&b+&<c+&d&d+&e&e
+&f+}
1000 aa(12)="o5r2{rrc+&c+&}{c+&c+&c+&d&e+&f+&a+&b+&<c+&d&d+&e&e
+&f+}
1010 aa(13)="r1r1
1020 aa(14)="o5{rg+}{rg+&}{g+&g+&g+&v11g+&v7g+&v13g+}2v15{rg}{r
g&}{g&g&g&v11g&v7g&v13g}2v15
```

▶今月の付録ディスクを使っている人の中に，絶対変な動きをさせている人がいると思います。私はいまワゴン車を隊列組ませて空中飛行させてます。　佐藤　夕子(21)神奈川県

```
 1030 aa(15)="o4{rrrrrrrrrrrrg+&g+&a+&b+&<d+&e&e&f+&g&}{g+r}
 1040 aa(16)="o4r2r1{rrrrrrrrrrrrg&g+&a+&<e+&f+&g&}{a+rf+r}
 1050 aa(17)="r2e+g+{a&a&aa+&}{a+&a+&a+&a+&a+&b&b+&<c+&c+}
 1060 aa(18)=aa(15)
 1070 aa(19)="o4r2r2.{g+a+<c+r}{c+&c+&c+&v12c+&v13c+v14c+&v15c+&
c+}1
 1080 aa(20)="{e+re+r}o4{rrrrrrrre+&e+&f+&a+&<c+&d+&e&e+}{e+re+
r}{rrrrrrrrrrg&g+&a&}{a&arr}r2.
 1090 aa(21)="|:r{g+&a+}r2r1r{g+&a+g+&a+}2r2r2.:|
 1100 aa(22)="e+2e+2<v12e+e+e+e+v15
 1110 aa(23)=aa(4)+aa(5)
 1120 aa(24)=aa(6)+aa(7)+aa(8)+aa(9)
 1130 aa(25)=aa(10)+aa(11)
 1140 aa(26)=aa(12)+aa(13)
 1150 aa(27)=aa(14)+aa(15)
 1160 aa(28)=aa(16)+aa(17)
 1170 aa(29)=aa(15)+aa(19)
 1180 aa(30)=aa(20)+aa(21)
 1190 aa(31)=aa(22)+aa(10)
 1200 aa(32)=aa(11)+aa(12)
 1210 aa(33)="o5r1r2{rrg+&a+}{b+&<c+&d+&e+}
 1220 aa(34)=aa(1)+aa(2)+aa(3)+aa(8)+aa(9)
 1230 for i=0 to 34:m_trk(4,aa(i)):next
 1240 aa(0)="@75q8o4v1514 r64 r
 1250 for i=0 to 34:m_trk(8,aa(i)):next
 1260 /* *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
 1270 /* *-*-*-*-*-*-*-*-*-*-*-ﾊﾞｯｸ1-*-*-*-*-*-*-*-*-*-*-*
 1280 aa(0)="@76q6o5v1214 r
 1290 aa(1)="|:{c+&c+&c+c+}r{rrc+r}{r>b+&b+r}<{c+&c+&c+c+}rr2:|
 1300 aa(2)="o3e+2d+2c+2>b+2 o4v15|:4{rra+r}{rra+r}    :|
 1310 aa(3)="l8q3v12o4 ra+ra+rg+rg+ra+ra+ra+ra+ra+ra+rg+rg+<rc+r
c+rc+rc+
 1320 aa(4)="rd+rd+rd+rd+>ra+ra+ra+ra+rarargrgra+ra+ra+ra+
 1330 aa(5)="{a+a+}{a+r}{a+r}{a+a+}{a+r}{a+r}{a+a+}{a+r}r2.a+r "
+aa(3)+aa(4)
 1340 aa(6)="l8o4q3ra+ra+ra+ra+ra+ra+ra+ra+
 1350 aa(7)="o5q7v12 b+g+d+g+d+g+b+g+ <c+>g+e+g+e+g+<c+>g+ <d+>a
+f+a+f+a+<d+>a+< c+1
 1360 aa(8)=">b+g+d+g+d+g+b+g+ <c+>g+e+g+e+g+<c+>g+ <c+1
 1370 aa(9)=">b+b+rrb+b+rrb+4r2.
 1380 aa(10)="l4v11|:4r{g+&g+&g+g+}{rrg+&g+}{g+rrr}r{g&g&gg}{rrg
&g}{grrr}:|
 1390 aa(11)="{a&a&ar}2{a&a&ar}2aaaa
 1400 aa(12)=aa(6)+aa(3)+aa(4)
 1410 aa(13)=aa(5)
 1420 aa(14)=aa(6)+aa(7)
 1430 aa(15)=aa(8)+aa(9)
 1440 aa(16)=aa(10)+aa(11)
 1450 aa(17)=aa(3)+aa(4)
 1460 aa(18)="|:4ra+ra+ra+ra+:|q8e+2d+2c+2>b+2 r2.<<a+
 1470 for i=0 to 18:m_trk(5,aa(i)):next
 1480 /**-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 1490 /* *-*-*-*-*-*-*-*-*-*-*-ﾊﾞｯｸ2-*-*-*-*-*-*-*-*-*-*-*
 1500 aa(0)="@76q6o4v814 r
 1510 aa(1)="|:{e+&e+&e+e+}r{rre+&e+}{rd+&d+r}{e+&e+&e+e+}rr2:|
 1520 aa(2)="o3e+2d+2c+2>b+2 o4q3v10|:4{rrc+r}{rrc+r}    :|
 1530 aa(3)="l8q3v12o4 rc+rc+r>b+rb+<rc+rc+rc+rc+rc+rc+r>b+rb+<r
e+re+re+re+
 1540 aa(4)="rd+rd+rd+rd+rc+rc+rc+rc+>rb+rb+ra+ra+<rc+rc+rc+rc+
 1550 aa(5)="{c+c+}{c+r}{c+r}{c+c+}{c+r}{c+r}{c+c+}{c+r}r2.c+r "
+aa(3)+aa(4)
 1560 aa(6)="l8o4q3rc+rc+rc+rc+rc+rc+rc+rc+
 1570 aa(7)="o4v12 >rb+rb+rb+rb+<rc+rc+rc+rc+rd+rd+rd+rd+>ra+ra+
ra+ra+
 1580 aa(8)="rb+rb+rb+rb+<rc+rc+rc+rc+>ra+ra+ra+ra+
 1590 aa(9)="<e+e+rre+e+rre+4r2.
 1600 aa(10)="l4v12o4|:4r{c+&c+&c+c+}{rrc+&c+}{c+rrr}r{c+&c+&c+c
+}{rrc+&c+}{c+rrr}:|
 1610 aa(11)=">{b+&b+&b+r}2{b+&b+&b+r}2b+b+b+b+
 1620 aa(12)=aa(6)+aa(3)+aa(4)
 1630 aa(13)=aa(5)
 1640 aa(14)=aa(6)+aa(7)
 1650 aa(15)=aa(8)+aa(9)
 1660 aa(16)=aa(10)+aa(11)
 1670 aa(17)=aa(3)+aa(4)
 1680 aa(18)="|:4rc+rc+rc+rc+:|q8e+2d+2c+2>b+2 r2.<<c+
 1690 for i=0 to 18:m_trk(6,aa(i)):next
 1700 /**-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 1710 /* *-*-*-*-*-*-*-*-*-*-*-ｷﾞﾀｰ-*-*-*-*-*-*-*-*-*-*-*
 1720 aa(0)="@77q8t145 r4
 1730 aa(1)="|:4r1:| o3e+2d+2c+2>b+2
 1740 aa(2)="@3o5v814r1r{e+ed+dc+c>ba+ag+gf+e+e+} @77o3v14 {e+e+
}{d+c+<a+g+}
 1750 aa(3)="@77|:6r1:|
 1760 aa(4)="o3v1414{e+e+}{aa+b+&b+}{d+d+}{gg+a+&a+}{c+&>a+&}a+2
r
 1770 aa(5)="|:8 r1:|
 1780 aa(6)=aa(4)
 1790 aa(7)="o5@76v 6c+1d+1
 1800 aa(8)="v918 d+d+d+d+d+d+c+d+e+e+e+g+e+2
 1810 aa(9)="d+d+d+d+c+4d+4e+1
 1820 aa(10)=aa(8)+">a+1
 1830 aa(11)="v6aar4aar4a4r2.
 1840 aa(12)="l4|:4r{a+&a+&a+a+}{rra+&a+}{a+rrr}r{a+&a+&a+a+}{rr
a+&a+}{a+rrr}:|
 1850 aa(13)="l8a4.ra4.ra4a4a4a4
 1860 aa(14)=aa(2)+aa(3)+aa(4)+aa(5)
 1870 aa(15)=aa(6)+aa(7)+aa(8)+aa(9)+aa(10)
 1880 aa(16)=aa(11)+aa(12)+aa(13)
 1890 aa(17)=aa(3)+aa(4)
 1900 aa(18)="l8v15|:o2a+2&a+a+<c+e+d+4.d+{e+&e+g+a+&}4a+4:|
 1910 aa(19)="o3e+2d+2c+2>b+2
 1920 aa(20)="r2.a+4
 1930 for i=0 to 20:m_trk(7,aa(i)):next
 1940 /**-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 1950 vsett()
 1960 m_play()
 1970 end
 1980 func vsett()
 1990 dim char v0(4,10)={
 2000 /*    AF  OM  WF SYC SPD PMD AMD PMS AMS PAN
 2010        8, 15,  2,  1,  0,  0,  0,  0,  0,  3,  0,
 2020 /*    AR  DR  SR  RR  SL  TL  KF MUL DT1 DT2 AME *bass
 2030       30, 18,  0,  6,  2, 33,  0,  9,  2,  0,  0,
 2040       31, 14,  4,  6,  5, 24,  0,  0,  5,  0,  0,
 2050       30, 20,  5,  7,  1, 31,  8,  0,  6,  0,  0,
 2060       31, 11,  3,  6,  2,  0,  2,  0,  0,  0,  0}
 2070 m_vset(73,v0)
 2080 dim char v1(4,10)={
 2090 /*    AF  OM  WF SYC SPD PMD AMD PMS AMS PAN
 2100       52, 15,  0,  0,  0,  0,  0,  0,  0,  3,  0,
 2110 /*    AR  DR  SR  RR  SL  TL  KF MUL DT1 DT2 AME *s.drum
 2120       31,  1,  3,  0,  0,  1,  0,  0,  2,  3,  0,
 2130       31,  1, 14, 13,  0,  3,  0,  1,  5,  0,  0,
 2140       31, 16,  0,  0,  0,  6,  0,  2,  1,  0,  0,
 2150       31,  1, 19, 13,  0,  3,  0,  0,  6,  0,  0}
 2160 m_vset(74,v1)
 2170 dim char v2(4,10)={
 2180 /*    AF  OM  WF SYC SPD PMD AMD PMS AMS PAN
 2190       11, 15,  2,  1,  5,  0, 15,  0, 23,  3,  0,
 2200 /*    AR  DR  SR  RR  SL  TL  KF MUL DT1 DT2 AME *trumpet
 2210       31,  0,  1,  6,  3, 20, 16,  6,  5,  0,  0,
 2220       12,  2,  4,  6,  5, 31, 18,  9,  7,  0,  0,
 2230       26, 12,  5,  7,  1, 30, 21,  9,  5,  0,  0,
 2240       23, 11,  3,  6,  2,  4,  0,  1,  0,  0,  0}
 2250 m_vset(75,v2)
 2260 dim char v3(4,10)={
 2270 /*    AF  OM  WF SYC SPD PMD AMD PMS AMS PAN
 2280       48, 15,  2,  1,200, 99,  0,  3,  0,  3,  0,
 2290 /*    AR  DR  SR  RR  SL  TL  KF MUL DT1 DT2 AME *back
 2300       31,  0,  5,  6, 21, 20, 10,  7,  5,  0,  0,
 2310       31, 25,  5,  6,  6, 12,  5,  6,  6,  0,  0,
 2320       21, 15,  0,  6,  6, 15,  6,  5,  2,  0,  0,
 2330       31, 26,  0,  8,  2,  0,  0,  1,  0,  0,  0}
 2340 m_vset(76,v3)
 2350 dim char v4(4,10)={
 2360 /*    AF  OM  WF SYC SPD PMD AMD PMS AMS PAN
 2370       48, 15,  2,  1,200, 99,  0,  3,  0,  3,  0,
 2380 /*    AR  DR  SR  RR  SL  TL  KF MUL DT1 DT2 AME *guiter
 2390       31,  0,  0,  2, 31, 26, 15,  7,  5,  0,  0,
 2400       31, 25,  0,  6,  2, 12,  2,  6,  7,  0,  0,
 2410       31, 15,  0,  6,  4, 15,  0,  5,  2,  0,  0,
 2420       31, 26,  0,  8,  2,  0,  0,  1,  0,  0,  0}
 2430 m_vset(77,v4)
 2440 dim char v5(4,10)={
 2450 /*    AF  OM  WF SYC SPD PMD AMD PMS AMS PAN
 2460       61, 15,  2,  1,200, 99,  0,  3,  0,  3,  0,
 2470 /*    AR  DR  SR  RR  SL  TL  KF MUL DT1 DT2 AME *vocal
 2480       17, 15,  0, 10,  0, 26,  0,  1,  0,  0,  0,
 2490       20,  8,  0, 15,  0,  0,  0,  1,  3,  0,  0,
 2500       20,  8,  7, 15,  0,  2,  0,  1,  1,  0,  0,
 2510       10,  0,  0, 15,  0,  0,  0,  1,  3,  0,  0}
 2520 m_vset(78,v5)
 2530 endfunc
```

リスト5　出たな!! ツインビー

```
10 MEM$(&HB1D8,36)=HEXCHR$("DB 70 41 43 01 00 25 23 11 1B 9C 5D
99 9F 86 00 00 02 45 45 85 0A F5 05 06 06 00 00 00 00 F3 C8 87 0
0 03 00")
20 MEM$(&HB2B0,36)=HEXCHR$("FA 00 10 31 50 31 1E 3C 19 1E 1E 1E
18 18 04 84 04 84 05 05 04 04 26 26 26 26 00 00 00 00 00 00 80 0
0 02 E1")
30 MEM$(&HB3AC,36)=HEXCHR$("FA 20 21 34 32 72 1D 6B 61 00 DE DF
5E 4D 01 01 01 82 00 00 00 00 11 15 15 06 00 00 00 00 00 C8 C6 0
0 02 E1")
40 MEM$(&HB43C,36)=HEXCHR$("FB 63 72 02 31 01 37 39 37 00 DF 4C
CF 8C 05 07 02 90 43 00 00 00 E5 F5 24 16 00 00 00 00 0C C4 88 0
B 02 00")
50 MEM$(&HB4F0,36)=HEXCHR$("FA 00 41 45 41 41 23 39 2E 01 0E 5F
0D 0F 09 11 0B 9F 00 80 00 00 29 CF 18 0A 00 00 00 00 F4 C8 80 0
0 02 00")
60 MEM$(&HB514,36)=HEXCHR$("FC 00 01 11 21 61 20 00 17 07 52 12
4F 14 0C 8A 0A 82 01 01 01 01 2A 3A 5A 3A 00 00 00 00 00 C8 80 0
0 02 00")
70 MEM$(&HB55C,36)=HEXCHR$("D3 00 2A 06 04 02 43 1B 33 00 1F 1F
1F 1F 18 10 0C 8A 40 C0 00 00 FC F8 F4 F5 00 00 00 00 0C C8 80 0
0 02 00")
80 MEM$(&HB634,36)=HEXCHR$("FB 00 71 06 03 3D 0C 1B 16 00 1E 1E
1F 12 04 01 14 8B 40 00 00 00 F1 F2 F3 F6 80 00 00 00 00 C8 80 0
0 02 00")
90 MEM$(&HB658,36)=HEXCHR$("F3 00 71 04 01 31 11 08 15 11 1A 1A
14 17 00 04 12 8B 40 C0 81 0C 0A 2C 3C 4E 00 00 00 00 05 C8 80 0
0 02 00")
100 MEM$(&HB67C,36)=HEXCHR$("FB 00 0E 06 07 00 11 18 16 00 1A 1A
 1A 16 04 08 16 90 40 40 80 00 32 72 BA F8 00 00 00 00 00 00 80
00 00 00")
110 MEM$(&HB6A0,36)=HEXCHR$("F9 20 05 00 07 03 00 14 0A 00 1F 1F
 1F 1F 1F 17 18 00 40 00 8C 0C 03 95 F8 07 00 00 00 00 00 00 80
01 03 E1")
120 MEM$(&HB6E8,36)=HEXCHR$("FC 00 07 05 01 01 02 00 00 00 1F 1C
 1F 5F 0B 90 13 8A 00 00 00 3F F5 F8 E6 C7 00 00 00 00 D3 C8 80
00 02 00")
130 MEM$(&HB70C,36)=HEXCHR$("83 00 08 0F 00 01 01 00 07 00 1E 1E
 1E 1E 1A 1C 0F 8F 00 DF 00 00 FD AE F8 F8 00 00 00 00 D3 C8 80
00 02 00")
140 TEMPO 0:PLAY "X"
150 PLAY "T160I16:I16:I3:I28:I9:I36:I34:V15:V11:V11:V11"
160 PLAY "V13L8O5A2.>C<B&B1B-2.B-A&A2L16F&F#&G&G#&A&A#&B&>C&L8C2
.FD&D1";
170 PLAY "[O5C1&C2L32C.&C#.&D&D#.&E.&F&F#.&G.&G#&A.&A#.&B&L4>C1&
C1";
180 PLAY "I25V14<<A2.>C<B-AGFGD8G4D8G2.GAB-.>C8&C2";
190 PLAY "D-.E-8&E-2C1&C1<A2.>C<B-AGFGD8GD8G2.GAB-.>F.B-2B-AGA8F
8C1&C2R";
200 PLAY "RI20V14O5L8F4E.D.CD4F4E.C.ER4F4E.D.CD.<B-.>DE.C.ER4A-4
G.F.E-";
210 PLAY "F4A-4G.E-.GRD-FA->C4<G4B-4F4G.A-.B-B.>D.<B>C#.<A.F#E.B
.>DF#.C#.<A";
220 PLAY "F2G2A-2B-2V13I16L4C<B2.&B1B-A2.&A1>C<B2.&B1>>D-C2.&C1]
:";
230 PLAY "V13L8O5F2.AG&G1G-2.G-F&F2<L32A.&A#.&B&>C.&C#.&D&D#.&E.
&F&";
```

```
240 PLAY "F#.&G.&G#&A2.>L8C<B&B1";
250 PLAY "[O4C1&C2L32C.&C#.&D&D#.&E.&F&F#.&G.&G#&A.&A#.&B&>L4C1&
C1";
260 PLAY "I26V12<RCF2.CF>C<B2.D1>DD-1<B-1A2A-2G1";
270 PLAY "RCF2.CF>C<B2.D1>D<F.B-.>F2.C2<F2.&F8>B-8A8F8C2.";
280 PLAY "RI20V14O5L8D4C.<B-.AB-4>D4C.<G.>CR4D4C.<B-.AB-.F.B->C.
<G.>C";
290 PLAY "R4F4E-.D-.CD-4F4E-.<B-.>E-R<B->D-FG4E-4F4D-4D-.D-.FE.G
.EF#.E.C#";
300 PLAY "<B.>E.G>C#.<A.F#D2E2F2G2I25V13R4";
310 PLAY "O4L32G.&G#.&A&A#.&B.&>C&C#.&D.&D#&E.&F.&F#&G4&G1<R4";
320 PLAY "F.&F#.&G&G#.&A.&A#&B.&>C.&C#&D.&D#.&E&F4&F1<R4";
330 PLAY "G.&G#.&A&A#.&B.&>C&C#.&D.&D#&E.&F.&F#&G4&G1R4";
340 PLAY "F.&F#.&G&G#.&A.&A#&B.&>C.&C#&D.&D#.&E&F4&F1I16]:";
350 PLAY "V12L8O2F4R1F4F4F4F4R1F4F4F4F4R1F4F4F4";
360 PLAY "[V13O4I25L1DEFGRRRRRRRRRRRRRRRR";
370 PLAY "V14L8O4I20R4A4G.F.EF4A4G.E.GR4A4G.F.EF.D.FG.E.GR4A-4B-
.A-.G";
380 PLAY "A-4>D-4<B-.G.B-RFA->D-E-4C4D-4C4<B-.>C.D-";
390 PLAY "D.E.DE.C#.<AG.>D.EA.F#.C#<B-2>C2D-2E-2I25V13O4L32";
400 PLAY "R4D.&D#.&E&F.&F#.&G&G#.&A.&A#&B.&>C.&C#&D4&D1<";
410 PLAY "R4C.&C#.&D&D#.&E.&F&F#.&G.&G#&A.&A#.&B&>C4&C1<";
420 PLAY "R4D.&D#.&E&F.&F#.&G&G#.&A.&A#&B.&>C.&C#&D4&D1";
430 PLAY "R4C.&C#.&D&D#.&E.&F&F#.&G.&G#&A.&A#.&B&>C2.O2V12I3F4R4
]:";
440 PLAY "V10O5L16F>CFC<F>CFC<F>CFC<F>CFC<G>DGD<G>DGD<G>DGD<G>DG
D<";
450 PLAY "G->D-G-D-<G->D-G-D-<G->D-G-D-<G->D-G-D-<F>CFC<F>CFC<F>
CFC<";
460 PLAY "F>CFC<F>CFC<F>CFC<F>CFC<F>CFC<G>DGD<G>DGD<G>DGD<G>DGD<
[V13I25";
470 PLAY "O3L1B->CDEV10O5I28L4A2.>C<B-AGFGD8GD8G2.GAB-B-8>C8&C2D
-D-8E-8&E-2";
480 PLAY "C1&C1<A2.>C<B-AGFGD8GD8G2.GAB-.>F.B-.F8B-AGA8F8C2&C8<B
-8A8F8C2.";
490 PLAY "RFL8E.D.CD4F4E.C.ER4F4E.D.CD.<B-.>DE.C.ER4A-4G.F.E-F4A
-4G.E-.G";
500 PLAY "RD-FA->C4<G4B-4F4G.A.B-B.>D.<B>C#.<A.F#E.B.>DF#.C#.<AF
2G2A-2B-2";
510 PLAY "I26V12O4L4FDI28V10O6GDGDGDI26V12O4D-CI28V10O6FCFCFC";
520 PLAY "I26V12O4FDI28V10O6GDGDGDI26V12O5G-FI28V10O6FCFCFC]:";
530 PLAY "V13O3L8FFFFFFFFFFFFFFFFFFFFFFFFFFFFFCDCFFFFFFFFFFFFFFF
F[";
540 PLAY "CCCCCCCCCCCCCCCCCCCCCCCCCCGCCC>C<CF4.FF4.FF4.F4CF4";
550 PLAY "F4.FF4.FF4.F4F4FF4.FF4.FF4.FF4.FA4.A-4A-4G4G4C4C4.F4.F
F4.F";
560 PLAY "F4.F4F4FF4.FF4.FF4.F4F4FF4.FF4.FF4.F4F4FF4.FF4.FFFCFRF
FF<";
570 PLAY "B-B-4B-B-B-4B-B-B-4B-B-B-4B-B-B-4B-B-B-4B-GG4GAA4AB-B-
4B->CC4C";
580 PLAY "D-D-4D-CC4C<B-B-4B->CC4CD-D-4D-E-E-4E-EE4EEE4EEE4EF#F#
4F#";
590 PLAY "CC4CCC4CCC4CCC4CFF4FFF4FFF4FFF4FFF4FFF4FFF4FFCDEFF4FFF
4F";
600 PLAY "FF4FFF4FFF4FFF4FFF4FFEDD-]:";
610 A$="AAAAAAAA":B$="EEEEEEEE":C$="P3<I3F4F4F4F4P1>I36":D$=">P1
I36AA"
620 PLAY "P3L8O3V12F4P1>I36"+A$+C$+A$+C$+A$+"P3<I3F4F4F4P1>I36["
;
630 PLAY "R1R1R1R1"+A$+A$+A$+"AAAAAAR4"+A$+A$+"V10I35>RE4R4E4RE4
R2.I36<V12";
640 PLAY A$+A$+A$+"AAAAAAAR"+A$+A$+A$+"AAAAR2I35>V10";
650 PLAY B$+B$+B$+B$+B$+B$+B$+"R1V12I36<L16A8AAA8AAA8AAA8AA";
660 PLAY "A8AAA8AAA8AAA8AAR1R1L8AAAAP3<I3G4"+D$+"P3<I3G4"+D$+"P3
<I3G4"+D$;
670 PLAY "AAAAP3<I3F4"+D$+"P3<I3F4"+D$+"P3<I3F4"+D$+"AAAAP3<I3G4
"+D$+"P3<I3G4";
680 PLAY D$+"P3<I3G4"+D$+"AAAAP3<I3F4"+D$+"P3<I3F4"+D$+"P3<I3F4R
4P1>I36]:";
690 PLAY "P2O5L4V15R1RDDDR1RDDDR1RDDD[DR2.R1DR2.R1DR2.R1R1R1R1R1
D.DR";
700 PLAY "DD8DD8DR8DR2.R1R1R1R1R1R1R1DR2.R1R1R1DR2.R1R1R1DR2.R1R
1R1";
710 PLAY "R2DRDRDR2.DRDRDR2.DRDRDR2.DRDRDR]:";
720 A$="I40CI39EI40C8C8I39E":B$="I40CI39EI40CI39E"
730 PLAY "O1L4S4,2,0,50=0"+A$+A$+A$+A$+A$+A$+"[";
740 PLAY A$+"I40CI39EE8I40C8I39E"+A$+"V16I37=3>L16P1GGG8P3FFF8P1
GGP3FFFP2<GGG";
750 PLAY "V15P3O1=0L4"+B$+B$+B$+"I40CI39EI40CI39E8E16E16"+B$+B$;
760 PLAY "I40CI39E8I40C8CI39E8I40C=3I37>V16P1F8F8P3<I40=0V15C8";
770 PLAY "=3I37>V16P2D8D8V15=0<P3I39E8E16E16"+B$+B$+B$;
780 PLAY "I40CI39EI40CI39E8E16E16"+B$+B$+B$+"I40CI39EE16E16E8E16
E16E8";
790 PLAY B$+B$+B$+B$+B$+B$+B$+"I40C8I39E8EI40C8I39E8E8E16E16";
800 PLAY B$+B$+"I40C8I39E8EI40C8I39E8EI40C8I39E8EI40C8I39E8E16E1
6E16E16";
810 PLAY "I40CI39E>I3V12GV15<I39E>I3V12GV15<I39E>I3V12G<V15I39E8
I40C8";
820 PLAY "I40CI39E>I3V12FV15<I39E>I3V12FV15<I39E>I3V12F<V15I39E8
I40C8";
830 PLAY "I40CI39E>I3V12GV15<I39E>I3V12GV15<I39E>I3V12G<V15I39E8
I40C8";
840 PLAY "I40CI39E>I3V12GV15<I39E>I3V12GV15<I39EE16E16E8E16E16E8
]:";
850 PLAY "S4,15,3,0=3L8O4K1AAAAAA>C<B4>F>F<F>F<F>F<F<B-4B-B-B-B-
B-A4";
860 PLAY ">F>F<F>F<F>F<F<AAAAAA>C<B4>F>F<F>F<F>F<F[D4.DD4.D4EERL
16";
870 PLAY "V13CDEFGAB->C<V11L8F4.FF4.F16&E16&DEERL16V13CDEFGAB->C
<V11";
880 PLAY "L8CCRCRCCCCCRCRCCC<BBRBRBBBBBRBRBBBB-B-RB-RB-B-B-B-B-R
B-RB-B-B-";
890 PLAY ">CC>C<CR>C<C>C<RCC4CC4.CCRCRCCCCCRCRCCC<BBRBRBBBBBRBRB
BB";
900 PLAY "B-B-RB-RB-B-B-B-B-RB-RB-B-B->FFRFRFFFFFRFRFFFL2DEFGFGD
CFGA-GFGFG";
910 PLAY "L16>BB-AA-GG-FEE-DD-C<BB-AA-GG-FEE-DD-C<BB-AA-GG-FEL4R
1R1";
920 PLAY ">C<B>GDGDGD<B-A>FCFCFCC<B>GDGDGDD-CFCFCFCL8]:";
930 PLAY "S4,15,3,0=3L8K1O4F1&FR2.RF1&FR2.RF1&FR2.R[";
940 PLAY "K2B-4.B-B-4.B-4>CCR>C2<D4.DD4.D4EE4>C2R2<C2&C2.C4D2.<G
1>D4F1";
950 PLAY "<<B-1>C4.C2C4>CC4CC4.R2C2&C2.C4D2.<G1>D4<B-4.>F4.B-2.>
C2";
960 PLAY "<F1<F1L2B->CDEDE<B-A>D-E-FE-D-E-D-E-R>C#<E>F#L8<<FF4F"
;
970 PLAY "GG4GA-A-4A-B-B-4B->L4AG<GRGRGR>GF<FRFRFR>AG<GRGRGR>B-A
<FRFRFRL8]:";
980 PLAY "S4,15,3,0=3L8K1O5F4R4DDD2F4F4F4F4R4DDD2F4F4F4F4R4DDD2F
4F4F4";
990 PLAY "[K1O4R2C2>G1R2<C2>G1";:FORJ=0TO1
1000 FORI=0TO1:PLAY "V10L32=0F.&F#.&G&G#.&A.&A#&B.&>C.&C#&D.&D#.
&E&";
1010 PLAY "F.&E.&D#&D.&C#.&C&<B.&A#.&A&G#.&G.&F#";:NEXTI
1020 FORI=0TO1:PLAY "G.&G#.&A&A#.&B.&>C&C#.&D.&D#&E.&F.&F#&";
1030 PLAY "G.&F#.&F&E.&D#.&D&C#.&C.&<B&A#.&A.&G#";:NEXTI
1040 FORI=0TO1:PLAY "A#.&B.&>C&C#.&D.&D#&E.&F.&F#&G.&G#.&A&";
1050 PLAY "A#.&A.&G#&G.&F#.&F&E.&D#.&D&C#.&C.&<B";:NEXTI
1060 IFJ=0THENPLAY "R1R1";
1070 NEXTJ:PLAY "F.&F#.&G&G#.&A.&A#&B.&>C.&C#&D.&D#.&EL16FC<AF>C
<AFCAFC<A";
1080 PLAY ">FC<AF>CAFCAFC<AV11=3K2L2R>EFGFGDCRGA-GFGFGL8B.>D.<B>
C#.<A.F#E.B.";
1090 PLAY ">DF#.C#.<AF2G2A-2B-2>L4C<BGRGRGRB-AFRFRFR>C<BGRGRGR>D
-C<FRFRFR]";
1100 A$="":B$=""
1110 PLAY "X"
```

(善）のゲームミュージックでバビンチョ

なんと，「ガンマ・プラネット」のBGM作曲者の三村隆之氏からハガキをもらってしまった。あの巧みなコード進行，ちょっとダークなバックに絡んでくる高速シンセソロ，「ガンマ・プラネット」のBGMって，私大好き。三村さん，暇つぶしにオリジナル曲書いてLIVE in宛てに送ってくださいよ。いや？

●EARY COLLECTION/古代裕三 CD:ALCA-328
アルファレコード　1,500円(税込)　7/21発売

いままでCDに収録されてなかった曲を集めたアルバム。X1用RPG「アルガーナ」(PSGバージョン)と「スキーム」(OPNバージョン)，X68000用アクションパズル「スライス」，そしてX68000芸術祭のテーマが収録されている。

お勧め度　8

●RAP a de LIC/DATA EAST CD:PCCB-00090
ポニーキャニオン　2,500円(税込)　発売中

先月「善バビ・対談コーナー」で登場いただいたGAMADELICによるアレンジバージョンオンリーのアルバム。タイトルからもわかるように，全編ラップ/ハウスアレンジが施されていて,かなりユニークなものに仕上げられている。単なるお遊びでない，とても音楽性が高いものになっており，デコのゲームを知らない人，和製ラップ音楽反対者にもオススメ。「ウォーク・ライク・スーパーコップ」はTMNを，「カルノフが街にやってきた」は坂本龍一をパロった感じになっており，ちょっとニヤリとさせられる部分も。

お勧め度　10

●PU・LI・RU・LA/TAITO　CD:PCCB-00091
ポニーキャニオン　1,500円(税込)　発売中

このゲーム大好きだった，なんかわけわからなくて。特殊魔法や効果音がキレまくっていたし，なんとも私の感性に合ったゲームだった。ゲームを知っているととても楽しいCDなんだけど，情景描写的な曲が多いので，逆にゲームを知らないとかなり聞きづらいかも。私が好きなのは，BGM I「ザックとメルの町」。とてもメロディアスでリズムが面白い。しかも，エンディング曲ではこのBGM Iがリプライズするところは泣かせるなぁ。

お勧め度　8

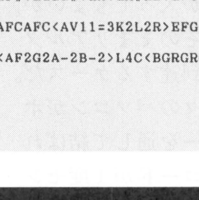
●矩形波倶楽部　CD:KICA 1020
キングレコード　3,000円(税込)　発売中

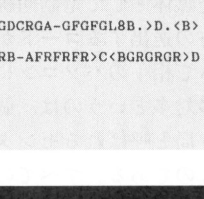
古川もとあき氏の「SOUND LOCOMORIVE」は先月紹介したが，以前にT-SQUAREの安藤マサヒロのプロデュースでオリジナルCDを出しているという報告が読者数人よりあったので，これを紹介する。今年年始に発売された「コナミ千両箱」の中のライブCDとほぼ同タイトルが収録されている。全曲フュージョン系のインストで，ギター中心の構成。ゲストミュージシャンがほとんどT-SQUAREのメンバーなのがなかなか。ベース：須藤満，ドラム：則竹，サックス：本田雅人……。

お勧め度　10

終わりに

さて，毎年恒例になりつつあるGAME MUSIC FESTIVALのお知らせ。今年も日本青年館で2日間かけて行われます。詳しいことはペンギン情報コーナーのINFORMATIONのところを見てね。んではまた来月。





パソコン通信に未来はあるか

Ogikubo Kei 荻窪 圭

今回は（も？）X68000からちょっと離れて，パソコン通信の話を取り上げます。パソコン通信の長所と短所，そしてメディアとしての可能性を，荻窪(はったり)圭先生がマジメに論究してみてくれました。

パソコン通信というものがある。

パソコンとパソコンが情報をやりとりする手段のひとつである。

当然それぞれのパソコンの外側には人間がいる。

我々が通常「パソコン通信」という場合，通信しているコンピュータの外側にいる人間をないがしろにはできない。

パソコン通信を解体する

1）「1対1」と「多対多」

パソコン通信には「1対1」か「多対多」しかありえない。

1対1というのは，個々のパソコンを結ぶ通信である。間に通信装置としてモデム，伝達媒体として電話回線が入ってもよいが，片方の送出するコードがそのままリアルタイムで相手のパソコンに到達するケースだ。

多対多というのは，個々のパソコンがホスト局と呼ばれるセンターを通して結ばれるものである。すべてのコードが1度センターを通すことにより，多対多の通信を可能にしている。

ホスト局を通す通信は，物理的に多対多をサポートするものであるが，論理的に1対1の通信を作り出すこともできる。それは，電子メールである。「ホスト局に送出されたコードは，特定のパソコンからしか読み出すことはできない」ということが，互いの信頼性の元に成り立っているシステムである。

我々が「パソコン通信」と称する場合，ほとんどは多対多のシステムを指している。いつしか，そうなった。

2）人と人とのコミュニケーション

パソコン通信という言葉には，パソコンどうしの通信という以上に意味は込められていない。

現実には，パソコンはただコードを吐き出すだけである。

すべての通信にはプロトコルが必要である。通信装置どうしの転送プロトコルだけでなく，互いに同じコード体系をもっていなければならない。しかし，転送プロトコルの点では厳密な一致が必要だが，転送するコードの体系は人間どうしの情報のやりとりとまではいかなくても，いくらか曖昧でも可能である。たとえば，PC-9801文字の存在が許されていることでもそれはわかる。ローマ数字や丸つき数字の存在だ。さらに，PC-9801のもつ漢字ROMのJISコードが1978年版であるのに対し，そのほかのほとんどのマシンが1983年版であることでもわかる。それでもなんとか成り立っているのは，人間が後ろに控えているからだ。

パソコン通信によって伝達されるのは，指定されたコードだけである。そこに意味はない。意味内容を付加するのは後ろに控えている人間である。

ときには，人間にとっては無意味なコードの転送も行われる。バイナリコードやISH化されたテキストである。バイナリコードは，グラフィックや圧縮されたテキストの場合，受信したコンピュータの側で復元してやれば，人間にとって意味のあるコードになる。プログラムの場合は，復元したものを受信側の人間が実行してはじめて意味のあるものとなる。

つまり，人間が関与しないコードはまず転送されないのである。

パソコン通信は機械と機械のコードのやりとりであるが，それぞれの機械の後ろには人間が控えており，やりとりされるコードは人間にとって意味のある形になってはじめて成り立つ。人間にとって意味のある情報を，機械で扱えるコードに変換して転送する。これがパソコン通信である。しかし，人間どうしのコード体系の取り決めや，その解釈法ははなはだ曖昧である。

3）テキストデータについて

パソコン通信でもっとも頻繁にやりとりされる文字コード体系は，多くはシフトJISと呼ばれる体系である(JISコードを使っているところもまれに存在する)。俗にいうテキストデータだ。

無論，デジタルデータであるから，転送によって意味内容が変質することはない。問題が生じるとすれば，送信者が伝えようと思った情報を的確にテキストデータに変換できたかという点である。さらに，受信者が送信されたテキストデータの意味内容を的確に理解したかという点にもある。

テキストコードは言語をもっとも単純化した形に変換したものにすぎない。それはあくまでも文字に割り当てられたコードを送受信するだけであり，そのコードを文字の形で視覚化して人間に提示するのはコンピュータの仕事である。文字がそのまま送られるわけではない。

つまり，コード体系はすべての機種で同じであるという前提があるが，視覚化された情報は機種によって微妙に異なる。Macintoshでは9ポイントや12ポイントのフォントとなる。X68000やPC-9801では16ドットのフォントとなる。ハイレゾマシンでは24ドットのフォントとなる。さらに，同じ16ドットでも機種によって書体は異なる。そういう意味である。

4）グラフィックコード

グラフィックの場合は，テキストコードのように受信する端から復元されるということはない。バイナリデータとして転送され，転送が終了したあとに受信側で復元される。

ここには，2つの関門がある。ひとつは，グラフィックデータを符号化するときの手法である。もうひとつは，復元する機種のグラフィック能力である。

前者では，データ圧縮が可能な手法が歓迎される。X68000ではPICであり，PC-9801ではQLDやMKI，256色データではGIFといった具合である。数は多く，たいていは自然発生的なユーザーレベルでの同意に基づく圧縮法である。GIFは，アメリカの大手

ネットワークCompuServeで標準と定められた規格である。最近では，フルカラーデータを圧縮/転送するためにJPEGが使われることもある。JPEGによる圧縮は，元の画像の情報量を落として行われることが多い。すると，700Kバイトあるデータが70Kバイトくらいにまで圧縮できる。復元された画像は多少画質が落ちるものの，それを解釈するのは人間である，という点においては大きな問題とはならない。

続いてグラフィック能力である。ひとつは表示できる色数。もうひとつはドットの縦横比という問題を抱えている。

色数に関しては，16色表示が標準のPC-9801が足を引っ張る格好になっている。おかげで，16色でいかに表現するかという些細な技術は発達したが，それは本質的な問題解決にはなっていない。

ドットの縦横比はX68000の512×512ドットの絵においてネックとなっている。

これらの問題はフリーウェアによって対処されているが，転送側と同じ画像を復元できているわけではないので，解決されているとはいえない。

5) 開かれているということ

パソコン通信は多対多対応である。論理的には不特定多数のパソコンによって引き出される情報であり，引き出された情報は不特定多数の人間によって参照される可能性をもっている。

後者についてはパソコン通信にかぎった問題ではないので，ここでは触れない。

問題は前者である。パソコン通信の世界では，ヒエラルキー的に権限が決められている。ここでは，大多数であるID取得者をベースに考える。

パソコン通信のホストには，公共の場としてBBSや電子会議室が設けられている。こういった場所への書き込みは，ID取得者全員によって参照されうる。ときにはゲストが参照できる場所もある。つまり，不特定多数（ないしは特定多数）の人間によって参照される可能性がある場である。これは，“外に向かって開かれた場所”である。

逆に，CUGというものがある。これはクローズド・ユーザーズ・グループの略で，多くの大手ネットワークがこの仕組みをもっている。クローズドというだけあって，特定のメンバーしか使うことのできない場所である。つまり，“閉じられた場所”である。

電子メールも同様に“閉じられた場所”である。

パソコン通信はメディアとして飛躍するか

いよいよ本題である。

画像の転送とかプログラムの転送，データベースの検索やお買物といったものもパソコン通信の醍醐味なのだが，考えてみたいのは，文字コードによるコミュニケーションとしてのパソコン通信なわけだ。画像にもプログラムにも，自分で作ってアップロードしたり転載したりした人の意志・意思が込められているものだが，それはあと回しで，まずは文字コードによるコミュニケーションの話である。

この手の話にはさまざまなテーマが見え隠れして非常に語るのがややこしい。しかし，本質的なレベルまで降りていくとかなり整理できるはずである。しかし，それを始めると，「人間のもつ言語処理能力」，「パソコン通信というメディアが人間に与える影響」というレベルさえ登場してしまう。私の頭が沸騰しそうな話題である。とほほ。

1) コミュニケーション

人と人とのコミュニケーションってやつは，生きていくうえで不可欠のものとなっている。しかし，それを成り立たせている約束事は明文化が不可能なはなはだ曖昧なものである。

面と向かったコミュニケーション。電話による声を使ったコミュニケーション。書簡によるコミュニケーション。コミュニケーションというと，こうした双方向ものが思い浮かぶが，「伝達」という意味では，マスメディアを使った発表（この原稿なんかもそうだ），講演，なんらかの作品の提示，といったものも含まれる。こういったところが一般的なものである。パソコン通信はそれに割り込んだまったく新しいものである。

アラン・ケイは，コンピュータはあらゆるメディアを模倣できるメタメディアだ，といった。つまり，およそデジタルデータに変換できるものなら，どんなものにでもなりうるのがコンピュータなのである。パソコン通信も同様だ。

パソコン通信は会話，書簡，マスメディアによる発表，作品の提示，講演，そういった伝達方法のもつさまざまな要素を併せもったメディアなのである。だからややこしい。送信者が会話の延長にあるものだという感覚でメッセージを送信することも可能だし，書簡だと思って送信することも可能だし，ボツのない投稿誌だと思って送信することも可能だ。人は未経験の新しい概念を作り出すことができないという前提において，パソコン通信でさえ，それぞれが既存のメディアのシミュレーション，あるいは既存のメディアの複合としてイメージしている。

たとえば，会話の延長と思っている者と，書簡の延長と思っている者と，マスメディアの一種と思っている者の間で，うまくコミュニケーションをとることが可能だろうか。しかも，伝達手段はきわめて情報が圧縮された文字コードのみによるものだけなのである。それ以前に，人はどこまで言語を自在に操れ，なおかつ，どこまで言語を解釈できるのか。そういった基盤もはなはだ曖昧で，『滅びゆく思考力』という本によると，近年ますます人々の言語処理能力は劣り，身振り手振りを交えたり，共通のバックボーンを支えにしたりしなければコミュニケートできなくなっているそうである。文字コードに書き手の表情はない。それを支えるのは共通のバックボーンである。

我々の言語処理能力はかなり不安なレベルであり，パソコン通信はそれに情報量の少なさが輪をかけていると思えばいいだろう。かといって，文字データ量を増やせばいいかというと，長い文章は読んでもらえず，それ以前に長い文章を論理的な破綻なく必要なことだけを誤解されることなく記述できる人自体，ごく少数にすぎず，そういう文章表現のための訓練さえ学校ではしないという状態だ。

さらに，あいつは何を考えてあの書き込みをしたのか，年齢はいくつか，本当に男なのか女なのか，それ以前に，人間なのかどうか。論理的には，同一人物が複数のIDをもって違う人物のふりをして書き込むことさえ可能なのだ。確認する手段をもたない。確認手段は相手の書き込みだけであり，それを信頼できるかどうかですべてが決まる。それは非常に脆い信頼である。脆いがゆえ，ひとたび疑心暗鬼に陥ると，とめどなく信頼性は失われ，泥沼化することになる。

2) ネットワークは仮想世界か

はなはだ曖昧な表現だが，「人格」というものがある。ネットワークは上記のように，はなはだ信頼性に欠ける空間である。そこでなおかつ人と人とのコミュニケーションを保とうと思えば，無理にでも信頼性をもたせねばならない。それには，書き込みから「人格」を読み取り，頭の中で文字コードに肉付けを行わねばならない。

それを選ばない人もいる。そういった人々は，ネットワークを仮想空間と位置付ける。現実世界とは独立した場であり，そこにいるのは自分の仮想人格である，という考えだ。これはひどく魅力的に思える。そう考える人々にとってネットワークは一種のゲームであり，現実ではない。ネットワーク内での出来事をほかの世界にもち込むと鬼の首を取ったように非難する。しかし，現実空間と仮想空間を厳密に分離することは可能だろうか。それ以前に，ネットワークは仮想空間だ，という定義が，現実として成り立つのか，という点については，仮想空間という発想に溺れて，考慮されていないようだ。

発想は面白いが，そうした前提をもつ空間は現実的に不可能だという結論を私はもっている。人間の意志が直接反映される場である限り，完全な仮想空間はありえない。あると考えていたら，それは能天気なのか，ゲームと間違えているかのどちらかである。ゲームをしたいのなら，「これはゲームである」と定義された場で行うべきだろう。

皮肉なことに，メンバーどうしの信頼性を高める手段としてもっとも有効なのが，オフラインミーティング，つまり，実際のメンバーと現実世界で会うことだ。

3) 匿名性の危険

パソコン通信によるコミュニケーションをややこしくしているもののひとつに，匿名性がある。匿名性，つまり，ハンドルネームという仮名のもとに参加できるため，実名ではできないような内容を平気で書けてしまう。現実世界にとらわれないために，現実世界では立場上できない話をするための匿名性というものもあるが，匿名性ゆえの暴力もまた存在する。前に述べたネットワークを仮想空間とみなす遊びも匿名であるがゆえにできたことだ。

匿名性には魔力がある。自分自身を安全な場においた無責任な発言が心理的に可能になるからだ。

しかし，いかに匿名によるものであっても，現実世界と仮想空間の完全な分離は不可能だ。ひとつの空間でひとつの名前という原則がある以上，匿名性に基づいた仮想人格を作ろうと思ったらそれを押し通さねばならなくなる。ネットワークを言語ゲームの場にすぎないと捉えるのならそれでもいいが，コミュニケーションの場として捉えるのなら，そうした仮想人格は足かせになるかもしれない。

私は，ネットワークにおいて，匿名は必然ではないと考えている。ハンドルネームという習慣は面白いものだし，否定はしない。実名公開主義のネットワークでもハンドルネームは使われている。

しかし，現実として，書き込みという表面化した当人の一部分と，実際の当人を混同してしまうような人がいるという面があることは否定できない。この原稿は，荻窪圭が書いたものであるが，荻窪圭＝この原稿なわけではない，というのと同じだ。この原稿は，荻窪圭の考えたり感じたり思ったりしたことの一部が，言語化というフィルタ（このフィルタが非常に曲者なのだ）を通して出てきただけのものである。

匿名性にはメリットとデメリットがあり，一概に否定はできないが，その魔力と暴力が可能だということは知っておくべきだろう。たとえば，実名では書けないようなことが平気で書けてしまいがちなこと（通常のコミュニケーションでは自分という存在を念頭において行われる。罵詈雑言を発すればそれがなんらかの形で自分に返ってくることを普通は考えてしまうが，仮名であればそれが直接自分に響くことがないような気がしてしまう）や，自分を安全な場所に置いたまま他人を攻撃できること（たとえば，仮名による実名攻撃は悪質といえるか否か）などである。そういった行為に及ぶか及ばないかは当人しだいであるわけで，ハンドルネームをもっていれば誰でもがやるわけではないのは当然である。

4) 共通のバックボーン

ネットワーク空間というのは，人と人とがコミュニケートするには，はなはだ脆い場である。と，そういう話を繰り返してきたわけであるが，現実には，多くのネットワークでコミュニケーションが成り立っている。うまくいくと，電話や会話や会合以上にコミュニケーションができている。そのへんが人間の面白いところであって，論理的信頼性がなくても，なんとかなってしまうのだ。

実際に会って話をしようが，電話で話をしようが，パソコン通信だろうが，結局我々が他人を判断するために行っているのは，経験に基づく類推にすぎない。多くの人と会った経験に基づいて，できるだけの情報を（身振り，表情，服装，顔，体型，声などなど）無意識のうちに読み取り，相手について類推する。パソコン通信ではその類推する手段が文字コードの並びだけしかない，というだけである。そうしたことがわかっていれば，その場に応じたコミュニケーションは成り立ちうる。特にテーマがはっきりした会議室であればなおさらだ。

しかし，それでも共通の言語的バックボーンは必要だ。言語的表現。日本語。しかし，それらもまた曖昧である。「灰色の雨」という表現がどれだけ普遍的なのか。文脈はどれだけ普遍的なものなのか。そういった内容は長い経験に基づいて判断される。何が通用して何が通用しないか。その判断は，育つうえでの環境に左右されることがらだ。

ある場で，会話を成立させるためのバックボーンがあまりにも狭い範囲でしか通用しないものであれば，そこは閉鎖的な雰囲気となる。広いものであれば，開放的な雰囲気となる。その判断は個々が下すものであり，その場の古い書き込みによって判断できる。

5) 歓迎する場

ほとんどのネットワークは「まず書き込め。みんなが歓迎するぞ」というメッセージに溢れている。その偽善的な香りさえするメッセージの本質はどこにあるか。確かに，その場のメンバーはそれぞれ「来るものは拒まず」な気持ちをもっているし，誰が来ても歓迎しようと思っている。しかし，現実になされるのは「まず書き込め。そうすれば，おまえがこの場にふさわしいかどうかがわかる」という意味ではないか。

しかし，他人に向かって「おまえはここにふさわしくない」と追い出すことはしづらい。そういうことができる人はそうはいない。実際には，さりげなく無視されたり，たしなめられたり，自ら雰囲気に合わないと判断して去っていったりする。それは現実世界のパロディのようなものである。場にそぐわない者はいづらい，というのはあらゆる場で起こることだ。

しかし，一方で，アクセス者を増やしたいという願望もある。はじめから「これこれこういう人のためのネットですから，そうでない人は来ないでください」といい切るのは難しい。アクセス者が減ることを何よりも恐れるからだ。

場によって歓迎される/されないが明らかに存在することと，開かれた場であることとは両立する，というのが重要だ。

パソコン通信はひとつのメディアであり，コミュニケーションの場である。建て前上「誰でも歓迎します」と書かれてはいても，コミュニケーションである以上，真に誰をも歓迎することはありえない。それでも，その場に触れたい場合はどうするか。パソ

コン通信では，読むだけという参加が可能である。それは非難すべきものではないだろう。読むものが書くものより多いというのはメディアとしてよく見られるものであり，なんの問題もない。パソコン通信の特長のひとつとして，読むだけの参加が可能なこと，が挙げられるくらいだ。

すると，参加者が限られてくる。誰もが知り合いといった雰囲気が出来上がる。新規参入者はますます入りにくくなる。閉鎖された雰囲気が強化される。参加者は，そこが閉じられた世界であるかのような勘違いをしてしまう。

閉じた世界のぬるま湯というものは確かにある。しかし，閉じていないのに閉じたかのような錯覚は危険である。

6) パソコン通信の可能性

特に重視されるべきは，草の根ネットワークではなく，大手ネットワークサービスである。1万人以上の会員を抱えうる大手ネットワークサービスでは，その中にさまざまな場を設けることができる。すると，パソコン通信独自のさまざまな性格付けをもった場を提供することが可能になる。

オンラインマガジンという試みもあるし(個人的にはむずかしい試みだと思っている。なぜなら，活字と違ってパソコンやワープロの画面の文字は粗くて読みづらいからだ。すると，CRTの文字向けの内容や文章の工夫，パソコン通信ならではの記事へのアクセスなど工夫すべきものが多すぎる)，オープンCUGという試みもある(ROMは誰にでもできるが，書き込みにはその場の責任者の許可が必要)。

人と人とのやりとりの面白さ，掛け合いの面白さと，偶然の面白さといったものが出てくれば，パソコン通信ならではの場が出来上がる。しかし，それにはある程度の言語処理能力をもつ人，ある程度の読解力をもつ人，提供できるだけのユニークな情報や考え方をもっている人，熱心な人が必要だ。となると，参加者の制限も考えざるをえない。

ネットワークは論理的に無法地帯である。誰が何を書いてもかまわないし，人の邪魔をしようが，他人のふりをしようがそれは可能である。しかし，それを「各自の良識」に頼ってばかりではなんの進歩もないし，ネットワークが広がることもないだろう。しかし，ネットワーク自体を「〜である」と定義することは不可能だ。ならば，1つひとつの場において，必要なルールを定めていくしかない。なんにしろ信頼性に基づかねばならないのは確かだが，雑誌における編集長的な，テレビや映画におけるディレクター的な存在をもつ場が出てこないと(たとえば,つまらないやつは参加禁止にするとか)，“読んで面白い場”というのは生まれないだろう。この，“読んで面白い場”がいちばん欠けているのである。いま，読んで面白いのは，興味の対象に対する情報が集積する場にすぎない。そういう情報交換の場だけでなく，あらかじめ読まれることを意識した作品の場が必要なのだ。ねるとんに誰からも指名されない女の子がいるように。

かつて，ネットワーカーたちがネットワークを自由の場だと感じ，すべては個人の裁量に任せ，制限はすべて忌むべきものとし，そういったことを踏まえたうえで参加していた。だから，制限のある場には拒否反応を示す人が多い。しかし，各自の良識に委ねる自由な場だけがネットワークではない。それもまたひとつの形ではあるが，ネットワーク内すべてがそうなる必要はないのだ。

パソコン通信の面白さのひとつとして，「年齢や職業を超えた人と知り合える」というのがもっとも教科書的で魅力的な意見としてもてはやされている。しかし，実際には，年齢や職業しか超えていない人たち，というういい方が可能なケースが多かった。

パソコン通信がコミュニケーションの場として成長していくには，オンライン状態での面白さが出てくる必要があるはずである。たとえば「いやあ，あのネットの××というヤツの書き込みが面白くてさあ。ついまた覗いてしまうんだよ」や，「あの会議室には面白いヤツが多くて」という面白さだ。オンライン状態での面白さというのは，オフラインミーティング，画像データやフリーウェアといったオフラインの状態で楽しむものを除いた，純粋に通信している間の面白さ。そういったものが必要なのである。

なんでこんな話になったんだ????

うーん。疲れた。なんでこんな話になったんだ。あ，そうだ。そろそろCommunication SX-68Kができる頃じゃないかとかまをかけて，パソコン通信の話でもしようと思ったんだ。それがこうなってしまった。最初はパソコン通信の気に入らないところをずらずら並べてやろうと思っていたのだが，つい客観的に書こうと思ったばかりに，客観視できない部分については触れられなかったのだな。いかん。もともとは，文章の上手い/下手，考えていることがユニーク/ありきたり，他人の迷惑を考える/考えない，被害者意識が強い/弱い，といった個々の個性は明らかに存在し，なかにはそういったなかから「読んで面白いものを書ける人」を集めた場が作れたら面白いのだが，そういうことをいうと平等ではないとか，選別意識があるとか，どういう基準で選別しているのか，といったことをいうヤツが絶対にいて，そのへんは非常にむずかしい問題ではあるが，だからといって平等を建て前に突き進むだけが能ではない，と，いうような話にするつもりだったのだ。

そもそも,ASAHIパソコンネットがいけない。あそこは実名主義だから，みんな実名が出る。だから，匿名性が併せもつ危険はなかった代わりに，妙にほのぼのとしていて，それが退屈でほとんどアクセスしていなかったわけだ。それが，「電脳筒井線」を皮切りに非常に面白くなってきた。電脳筒井線終了後も，参加者の個性を生かしたユニークな場をどんどん作っている。面白そうだから，ついニタニタして参加してしまう。それを踏まえてしまったわけである。

どうやって面白い個性を引き出すか，面白い個性を見つけたら，どうやってそれを発揮する場を与えるか，ネットワークを独立した世界ではなく，現実世界とどう絡めていくか，そういった試みがいままでなされなかったのが不思議に思えるくらいだ。書き込む人が工夫をすれば，常に読まれることを意識して書くようにすれば，まだ外に向かって面白い場は作れるのである。そして,「すべての人が平等に読み書きできるだけがネットワークの面白さではない」ことが発見できたのだ。

本当は，情報を収集できるとかフリーウェアがあるといった利害を超えた，参加しているパーソナリティの面白さでさらに人を集められるようなネット，なんの得もしないけれど面白いからついアクセスしてしまうネットが出てくるといいと思っている。

で，このままCommunication SX-68Kが完成すれば，来月はパソコン通信の話に突入する(じゃあ，今月はなんだったんだ?)。完成しなければ，FIXER ver.4.0のバージョンアップの話になるんじゃないかな。

今回はちょっと読む人も大変だったろうけど，たまにはこういうのもいいでしょう。X68000の話もなかったし，ちょっとだけ。私はMuTerm 1.23を使って通信していて，ログの整理にはmicroEMACSを使っているよん。ってところでどうでしょうか。

★(影)のショートプロぱーてぃ その35

夏です，金鳥です，花火です。

Kageyama Hiroaki 影山 裕昭

ああ暑い暑い。夏なんだから当たり前だけど。さて，今月は長い休みを持て余している人にオススメの素数計算プログラム「SOSUU.X」と，気分だけでも涼しくさせてくれる「HANABI.BAS」です。さあ，頑張って夏をのりきりましょう。

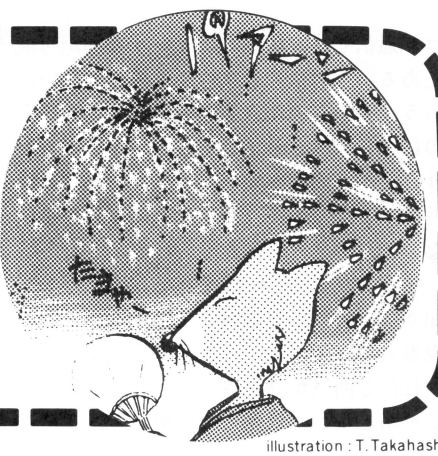
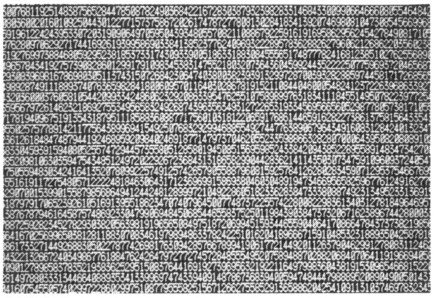


illustration : T.Takahashi

さぁて，学生諸君には長い長い夏休みの到来です。社会人の方はお勤めご苦労さまです。私も来年は社会人になる予定ですが，そのための就職活動もやっと終わりました(ちなみにコナミにも行ったけど適性検査で落ちたみたい。ダサダサ)。この本が出る頃には，もうすっかり落ち着いているハズです。大学は文系でゼミも卒論もないし，卒業まで毎日がもう休みのようなもの。

そのことを社会人の友人に話すと「学生のうちに遊んでおけよ」ってしきりに忠告してくれるんですねぇ。でも，実際に学生の間に毎日遊んだからって社会人になってから遊ばなくても平気かっていうと，そんなバカなことはないよね。“遊び溜め”がきかないからね。それでも忠告は忠告として受け入れて，いまの間に遠出だけはしておこうと思うんだな。いわゆる旅ですよ，旅。かっこいいよなぁ，旅って響き。

旅といえば，会社から1カ月休暇をとって宿の予約もせずにインドへひとり旅した友人がいて，そいつはもともと危ない奴なんだけど，噂では現地で薬（バファリンぢゃないぜ）が癖になって日本にいくらか持ち帰ったという話。あくまで噂ですけどね。それって税関通るとき勇気いるよなぁ。私なんかやましいことがなくても，税関通るときは緊張したもん。そういえば(て)氏も旅に出ているけど，いったいどこにいるのやら。元気でやっているのでしょうかねぇ。きっと彼のことですからどこにいても腹が減ったら牧草を食べ，喉が乾いては酒を飲み，カラオケをやっていることでしょう。



SOSUU.X

そんなこんなで，私も旅することにしました。といっても温泉旅行ですけど……。目的地まで車で3時間くらいだし，行くメンバーの中で学生は私だけということで，いまだからできるというわけではないんだけど，周りのやつらは帰ってきてから次の日は朝から仕事でしょ。それに比べりゃ私なんて昼ごろ起きて，夜になったらこの原稿を編集室に持っていって，ストII'を朝までやって……。ああ気楽，気楽。あと半年間，楽しませていただきましょう。

素数への挑戦

身の上話はこれくらいにしておいて，今月も2本のショートプロを紹介しましょう。6月号に夜寝る人のBASICとして，nのm乗を計算するBASICプログラムが掲載されましたのは記憶に新しいと思います。今月の1本目は先日発見された最大のメルセンヌ素数をX68000に計算させてしまおうというストロングなプログラムです。

SOSUU.X　for X68000

（要X-BASIC，アセンブラ，リンカ）

大阪府　野崎　哲也

素数とは，1以外の正の整数で1と自分自身以外に約数を持たない数のこと。たとえば2，3，5，7，11などです。すでに紀元前300年頃の数学者のユークリッドさんが，素数が無限に存在することを証明しているんですね。素数というのは，円周率と同じくらい数学の世界では熱心に研究されてるらしいです（わたしゃよく知らん)。それで野崎さんの原稿によると，いままで最大のメルセンヌ素数は2の86243乗マイナス1だったそうですが，先日新たに発見されたメルセンヌ素数は2の756839乗マイナス1だそうです。うーん，いったい何桁になるんだ。えーっと，227832桁ですか。数字を読み上げるときはどういえばいいのか困ってしまいますねー。

プログラムはマシン語(SOSUU.X)とX-BASIC (SOSUU.BAS) の2本があります。まず，リスト1のマシン語プログラムをED.Xなどのエディタを使って打ち込みます。ファイル名はSOSUU.Sです。なお行番号は説明のためにつけたもので，打ち込む必要はありません，念のため。打ち込んでディスクに保存したら，

A>AS SOSUU

でアセンブルしてから，

A>LK SOSUU

としてSOSUU.Xを作成します。

このプログラムは計算結果をディスクに保存するので，空き容量が少ないとプログラムの実行ができません。SOSUU.X実行前にディスクの空き容量が最低500Kバイト以上あれば問題ないでしょうから，その点を確認することを忘れないでください。

実行手順は，まずSOSUU.Xを実行して2の756839乗を計算させます。10MHzのX68000で約30時間の計算時間を必要としますが，30時間もの間X68000を占有されてしまうのは辛いですよね。そこは野崎さんも感じていたようで，実行途中にシフトキーを押すと，その時点までの計算結果をディスクに保存することができ，あと何乗の計算が残っているか画面上に報告されるようにプログラミングされています。また，SOSUU.Xをシフトキーを押しながら実行すると(A>SOSUUと打ち込んでシフトキーを押しながらリターンを押す）ディスクに保存してあるデータをロードして，そこからの計算を再開することができます。SOSUU.Xを実行中にX68000でほかの仕事をしたくなったら，いま述べた方法で対処してください。

計算が終わるとSOSUU.DATというファイルが作られます。このファイルはそのままではTYPEコマンドやエディタで見る

ことはできません。それを見られるように変換するためのプログラムがSOSUU.BASです。X-BASIC上からSOSUU.BASを実行すると，SOSUU.DOCというファイルができます。TYPEコマンドなりエディタでSOSUU.DOCを見て，その桁数の大きさに驚いてください。プログラムは2の累乗を計算しているだけなので，実際のメルセンヌ素数はSOSUU.DOCの内容から1を引いたものです。自分でSOSUU.DOCを書き換えてくださいね。

さて，投稿原稿によると「アセンブラはZ80の知識と，アセンブラマニュアルを適当に読んだだけで組んだので，かなりいい加減なので深く追及しないでください」とありますが，あえてひとつ2つ言及しておきましょう。

たとえば27行に，

```
    move.l  (mm1),d6
```

とあります。これは219行のmm1番地から1ロングワードの内容をd6レジスタに転送する命令です。68000の場合，mm1はプログラマが付けたラベル名で，アドレスレジスタではありませんから( )は付ける必要がなく，

```
    move.l  mm1,d6
```

と書くほうが一般的だと思います。d6レジスタの代わりにHLレジスタを使ってZ80のマシン語で同じことをやるなら，

```
    LD  HL,(mm1)
```

と書くので，野崎さんは( )を付けたのでしょうね。Z80から68000のアセンブラへ移るときは，見た目のデータの転送方向が逆になることと絶対アドレスの記述形式が変わってくるので注意が必要です。それからSOSUU.Xを作成するとソースリストの長さのわりに実行ファイルがバカでかいので不思議に思うかもしれません。これは223，240行で101260バイトものワークを確保しているからです。初期値なしのワークエリアを確保する場合は，その前に.bssを書いておくことを勧めます。これだけでソースがだいぶ小さくなるし，アセンブル速度も速くなりますよ。.bss命令を一度指定し，あとで再び初期値付きのワークを確保したい場合は，再度.dataを記述しなければなりません。SOSUU.Sの場合は223行以降が，

```
        .bss
buff:
        ds.l    size
        .data
buff2:
        dc.l    1
```

のようになるでしょう。初期値付きのワークと初期値なしのワークは，分けておくとソースリストを見やすく書けます。

ところでメルセンヌ素数とただの素数の違いってなんですか？　わたしゃ数学屋じゃないもんで，よくわかりません。野崎さんの報告を待って次号以降で補足できればと思っています。

夏の風物詩といえば……

夏の風物詩といえば金鳥と花火ですよね。というわけで，今月の2本目はX68000のディスプレイの中に花火を打ち上げてしまおうというプログラムです。う～ん，夏だなあ。

HANABI.BAS for X68000
(要X-BASIC，Cコンパイラ)
群馬県　杉本　達也

プログラムはX-BASICで書かれていますが，BAStoCでコンパイルされることを前提にしているので，実行にはCコンパイラが必要です。リスト3（HANABI.BAS）を入力して，

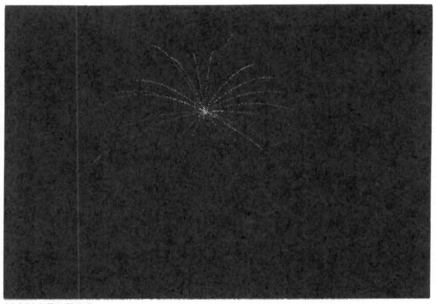
```
    A>CC /W /Y HANABI.BAS
```

としてください。打ち込み間違いがなければHANABI.Xが作成されます。もしエラーが出たらリスト3が正しく入力されているか確認して再度コンパイルしてください。

うーん，頑張って花火の雰囲気を出していますねー。これは「しだれ柳」っていう花火ですか？　部屋の明かりを消してみると，真っ暗な部屋の中にサーッと光の尾を引いていく感じがよくわかりますね。これで花火の消え方にもうひと工夫あれば，もっとよかったんではないでしょうか。

投稿原稿によれば「ギャラガ'88（電波新聞社）を遊んでいたら自分でも花火を上げてみたくなった」とありますね。いいですねー，その心意気。何事もできないと思ってやらなければ，いつまでたってもプログラムなんか組めるようになりませんからね。その調子で今後もチャレンジ精神を忘れないで頑張ってください。

そんなわけですから，投稿されてきたプログラムは花火の爆発音にギャラガ'88のPCMが使用されていたのですが，掲載にあたってその部分を削除させてもらいました。ギャラガ'88をお持ちの方で効果音をつけたいという方のための変更点を紹介しておきます。



HANABI.BAS

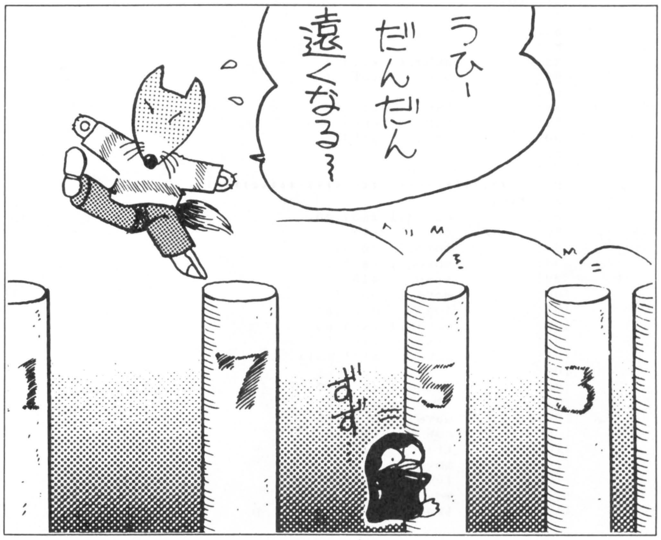
追加部分

```
  35 char adata(4434) : int fp
  62 fp=fopen("g8v107.pcm","r")
  63   fread(adata,4434,fp)
  64 fclose(fp)
 355 a_play(adata,3,3)
```

以上がHANABI.BASの追加点です。つぎにギャラガ'88のディスク1にあるドキュメントを参考にして，PCMデータを分解してください。その中のg8v107.pcmというファイルをHANABI.BASと同じディレクトリにコピーします。これで準備は完了です。改造したHANABI.BASをコンパイルして実行すると花火の爆発音が鳴るようなって，さらによくなるぞー。

画面に蚊とり線香の渦巻きを描いておいて，時間とともに蚊とり線香が灰になっていって，それで経過時間がわかるような環境プログラムがあったら面白いだろうね（そんなこと思うのは私だけか）。

てなところで今月はおしまい。来月は私が担当する最後の月の予定です（予定は未定という話もあるけど）。（で）ファンの皆様には申し訳ありませんが，あと1カ月我慢してお付き合いくださいね。それでは，また来月。



リスト1 SOSUU.S

```
************************************
***   素数を求めるプログラム      ***
***                    、          ***
***                    '           ***
***       平成４年４月５日         ***
**  PROGRAMMED   By  野崎  哲也 **
************************************
*
*この数字が、何乗するかです。以下、nで表します。
*
rui     equ     756839
*
* n / log(10) * log(2) / 9 を越える最小の整数
*
size    equ     25315

main:
        bsr tim
        move.l  #1,d6
        move.l  #rui,d7

        moveq.l #2,d0
        trap    #15
        btst    #0,d0
        beq     lo1
        bsr     load
        move.l  (mm1),d6
        move.l  (mm2),d7

        move.l  d7,d0
        bsr     pri
        bsr     pri

lo1:
        move.l  d6,(mm1)
        move.l  d7,(mm2)

        moveq.l #2,d0
        trap    #15
        btst    #0,d0
        beq     loa1
        move.l  d7,d0
        bsr     pri
        bsr     tim
        bsr     save
loa1:

        move.l  (mm1),d6
        move.l  (mm2),d7

        move.l  d7,d0

        move.l  d6,d5
        clr.l   d3
        lea     buff2,a6

lo2:
        move.l  (a6),d4
        add.l   d4,d4
        add.l   d3,d4
        sub.l   #1000000000,d4
        bpl     lo3
        add.l   #1000000000,d4
        clr.l   d3
        bra     lo10
lo3:
        move.l  #1,d3

lo10:
        move.l  d4,(a6)
        subq.l  #4,a6

        subq.l  #1,d5
        bne     lo2

        move.l  d3,d0
        beq     lo4

        move.l  #1,(a6)
        addq.l  #1,d6
lo4:



        subq.l  #1,d7
        bne     lo1

        bsr     tim
        bsr     save
        dc.w    $ff00

        .even
****************************************
tim:
        moveq.l #$54,d0
        trap    #15
        move.l  d0,d1
        moveq.l #$55,d0
        trap    #15
        move.l  d0,d1
        lea     date,a1
        moveq.l #$5a,d0
        trap    #15
        move.b  #' ',(a1)

        moveq.l #$56,d0
        trap    #15
        move.l  d0,d1
        moveq.l #$57,d0
        trap    #15
        move.l  d0,d1
        lea     time,a1
        moveq.l #$5b,d0
        trap    #15
        move.b  #' ',(a1)

        pea     prin
        dc.w    $ff09
        addq.l  #4,sp

        rts
**********************************
pri:
        bsr     change
        lea     buf,a1
        moveq.l #$21,d0
        trap    #15

        lea     m3,a1
        moveq.l #$21,d0
        trap    #15

        rts

*****    Ｄ０．Ｌを１０進数の文字列に変換
change:
        move.l  d0,d1
        move.b  #10,d3
        lea     buf1,a0

loop:
        move.l  d1,d0
        beq     zeroo
        clr.l   d1

loop1:
        addq.l  #1,d1
        sub.l   #10,d0
        bge     loop1

        subq.l  #1,d1
        add.l   #10,d0
        add.l   #'0',d0
        move.b  d0,-(a0)
        bra     jumpp
zeroo:
        move.b  #' ',-(a0)
jumpp:
        subq.b  #1,d3
        bne     loop

        rts
******************************************
save:
        move.w  #$20,-(sp)
        pea     nameptr
        dc.w    $ff3c
        addq.l  #6,sp

        move.l  d0,d2
        move.l  #size*4+12,-(sp)
        pea     mm1
        move.w  d2,-(sp)
        dc.w    $ff40
        add.l   #10,sp

        move.w  d2,-(sp)
        dc.w    $ff3e
        addq.l  #2,sp
        rts
******************************************
load:
        move.w  #0,-(sp)
        pea     nameptr
        dc.w    $ff3d
        addq.l  #6,sp

        move.l  d0,d2
        move.l  #size*4+12,-(sp)
        pea     mm1
        move.w  d2,-(sp)
        dc.w    $ff3f
        add.l   #10,sp

        move.w  d2,-(sp)
        dc.w    $ff3e
        addq.l  #2,sp

        rts
******  ワーク・エリア

prin:
        dc.b    13,10,'ただ今は'
date:
        dc.b    '
time:
        dc.b    '          です。',13,10,0
        .even

buf:    dc.b    0,0,0,0,0,0,0,0,0,0
buf1:   dc.b    0

m3:     dc.b    '回です。',13,10,0
        .even
nameptr:
        dc.b    'SOSUU.DAT',0

        .even
mm1:
        dc.l    0
mm2:
        dc.l    0
buff:
        ds.l    size
buff2:
        dc.l    1

```

リスト2 SOSUU.BAS

```
 10 /*  140行と150行とを削除すると改行しません
 20 str d
 30 char a(8)
 40 int dat(1)
 50 int f1,f2,i=0,j
 60 f1=fopen("sosuu.dat","r")
 70 if f1 then print "ファイルが見つかりません。":end
 80 f2=fopen("sosuu.doc","c")
 90 fread(a,8,f1)
100 repeat
110 fread(dat,1,f1)
120 until dat(0)
130 repeat
140 i=i+1
150 if i=11 then fputc(13,f2):fputc(10,f2):i=1
160 d=right$("00000000"+str$(dat(0)),9)
170 for j=1 to 9
180 fputc(asc(mid$(d,j,1)),f2)
190 next
200 if feof(f1) then fcloseall():end
210 fread(dat,1,f1)
220 until 0
```

リスト3 HANABI.BAS

```
 10 /*
 20 /* 花火.bas        By Sugimoto tatsuya
 30 /*
 40 int i,j,k,r,x,y,y1,y2,col
 50 int c(36),s(36),e(36),x_zahyou(5040),y_zahyou(5040)
 60 float d(36)
 70 /*
 80 screen 2,0,1,1:console 0,31,0
 90 randomize(val(right$(time$,2)))
100 /*
110 for i=0 to 35
120   c(i)=cos(i*11*pi()/180)*2*100
130   s(i)=sin(i*11*pi()/180)*2*100
140 next
150 /*
160 while inkey$(0)=""
170   init()
180   x=rnd()*384+192       /* 花火の出現X座標(192～575)
190   y=rnd()*100+100       /* 花火の出現Y座標(100～199)
200   v=int(rnd()*3)-1      /* 花火の上がる方向 -1=左 ,0=真上,1
=右
210   r1=int(rnd()*50)+90  /* 花火の半径は 90～139
220   j=512-y+40
230   for i=0 to r1
240     make_hanabi(i,i)
250     y1=512-abs(sin(i*pi()/180))*j
260     y2=512-abs(sin((i+1)*pi()/180))*j
270     line(x,y1,x+v,y2,13)
280     line(x-v,512-abs(sin((i-1)*pi()/180))*j,x,y1,0)
290     x=x+v
300   next
310   line(x-v,y1,x,y2,0)
320   y=y2+40
330 /*
340 r=int(rnd()*5)*30
350 col=int(rnd()*3)*2+5
360 for j=r/30 to r1
370   for i=0 to 35
380     pset(x+x_zahyou(r),y+y_zahyou(r),col)
390     r=r+1
400   next
410   for k=1 to r:next
420 next
430 /*
440 for i=0 to 419
450   for j=0 to 11
460     pset(x+x_zahyou(i+j*420),y+y_zahyou(i+j*420),0)
470   next
480 next
490 /*
500 endwhile
510 end
520 /*
530 func init()
540   for i=0 to 35
550     d(i)=-40
560     e(i)=int(rnd()*2)+101 /* 101,102
570   next
580 endfunc
590 /*
600 func make_hanabi(a;int,b;int)
610 int i,j
620 j=a*36
630 for r=a to b
640   for i=0 to 35
650     x_zahyou(j)=r*c(i)/100
660     y_zahyou(j)=r*s(i)/100+d(i)
670     d(i)=(d(i)+1)*e(i)/100
680     j=j+1
690   next
700 next
710 endfunc
```

(影)のぱーてぃハンズ

ぶれちゃイヤイヤ

さて，ぱーてぃハンズも，(で)氏が2カ月後に帰ってくるとすると，今月と来月の2回しかないわけですよね。そんなわけで，何かしらプログラムを作成して，ねっとりとした解説をすることはさっぱりとあきらめてしまっのであった。まあ，自己中心的でわがままなB型ですから。で，代わりに何をやろうかと考えてみたんですが，結局何も考えつかなかった。ハハハ。

しかし何かやらなきゃならないということで，以前に質問箱で掲載したことのあるvdisp.fncを再掲載します。X－BASICでアクションゲームを作ったことのある人ならわかるでしょうが，画面をスクロールさせたりキャラクタの移動をさせようとすると画面がぶれるときがあります。インタプリタ上では目立たなくても，コンパイルして実行したらぶれまくっていた，なんてこともあります。これは画面の書き換えを垂直帰線期間内に行うことで回避できるのですが，X-BASICにはそのための命令が用意されていません。このプログラムはX-BASICで垂直帰線期間の検出を行うための外部関数です。vdisp.sを入力してアセンブル，リンクしてできたvdisp.xをvdisp.fncにリネームしたらbasic.xと同じディレクトリにコピーしてください。次にエディタなどを使ってbasic.cnfに，

func = vdisp

を追加してください。これで関数vdispが使えるようになります。vdisp(0)で垂直帰線期間まで待ち，vdisp(1)で垂直表示期間まで待ちます。ライブラリもこのプログラムを若干変更するだけで作成することができます。その説明は来月号のハンズで。それまでvdisp.fncを作成しておくように。それでは，また来月。

リスト

```
*
* vdisp(0) で 垂直帰線期間まで待つ
* vdisp(1) で 垂直表示期間まで待つ
*
GPIP:              equ       $e88001

          .include           iocscall.mac

* infomation table

          .dc.l    x_init
          .dc.l    x_run
          .dc.l    x_end
          .dc.l    x_sys
          .dc.l    x_brk
          .dc.l    x_ctrl_d
          .dc.l    x_res1
          .dc.l    x_res2
          .dc.l    ptr_token
          .dc.l    ptr_param
          .dc.l    ptr_exec
          .dc.l    0,0,0,0,0

x_init:
x_run:
x_end:
x_sys:
x_brk:
x_ctrl_d:
x_res1:
x_res2:
          rts

ptr_token:
          dc.b     'vdisp',0
          dc.b     0

          .even
ptr_param:
          dc.l     vdisp_par
vdisp_par:
          dc.w     2
          dc.w     -1        * 戻り値なし
ptr_exec:
          .dc.l    vdisp

          .text
vdisp:
          tst.l    12(sp)
          bne      vdisp4
vdisp2:
          lea.l    GPIP,a1
          IOCS     _B_BPEEK
          btst.l   #4,d0
          beq      vdisp2  * 帰線なら
vdisp3:
          lea.l    GPIP,a1
          IOCS     _B_BPEEK
          btst.l   #4,d0
          bne      vdisp3  * 表示なら
          clr.l    d0
          rts

vdisp4:
          cmpi.l   #1,12(sp)
          bne      error

          lea.l    GPIP,a1
          IOCS     _B_BPEEK
          btst.l   #4,d0
          bne      vdisp4  * 表示なら
vdisp5:
          lea.l    GPIP,a1
          IOCS     _B_BPEEK
          btst.l   #4,d0
          beq      vdisp5  * 帰線なら
          clr.l    d0
          rts

error:
          lea.l    err_mes,a1
          moveq.l  #1,d0
          rts

          .data

err_mes:
          dc.b     '既定外の引数です',13,10,0
          .even

          .end
```

みーん
みーん
KYOKO IN ANOTHER CG WORLD
フリーの生活って
こんなんだすの巻 その1
きーん
わたしの脳ミソは容量があんまし大きくない。でもって、大切なことはぜんぶ手帳にメモっておきます。
シャープの電子手帳も使っているよ…
6月も終わりのころ…
FILOFAX
どれどれ
ぱっ
メモ
Oh!Xメモ
とにかくメモ
がびーーーん
どーしよどーしよどーしよー
まえからわかってたくせに…

今回のCGデータ

総物体数　643

うちメタボール数
290

光源　5

1280×1024
ピクセル

1670万色フルカラー
を４×５ポジで
出力

使用ソフトは
C-TRACE
サイクロン

第72回

猫とコンピュータ

新説・猫×7×0.7

Takazawa Kyoko
高沢 恭子

「創刊10周年」の文字を見て，連載当初をちょっとだけ思い出したキョウコさん。慣れないパソコンを使いはじめ，パソコン通信を覚えたばかりのあの頃を懐かしく語ってくれました。

「おばあちゃんきてたの？」

霧雨が降ったりやんだりしている午後のこと，新宿のおばあちゃんがおとずれていた。学校から帰ったトオルの傘が少しだけぬれている。制服が夏姿になると，あとから梅雨がやってくる。

明るい紺色の傘をトオルがリビングの窓ぎわに広げたら，すかさずホンニャアが中にすわった。小さなドームやアーチを見つけると，猫としてはまず自分が入って検分しなくてはならない。

平成年齢計算法

「Oh!MZ」の時代から10周年と聞いて，ホンニャアとの歳月を思い返す。

いっしょに生まれた妹猫と引き離して，ホンニャアだけをもらってきた雨の日。連載をはじめる少し前のことだった。妹猫は左右の目の色が違って，ゴールドとブルーだった。そういう目は，「オッドアイ」とか「金目銀目」と呼ばれて珍重されるのだそうだが，青い目のホンニャアだけを飼うことにしたのだった。

傘の中でホンニャアは満足そうだ。

「あの猫はネ，私に興味があるのよ」

おばあちゃんが言った。

「へぇー，そうなの？」と私。

「私はアナタみたいに用もないのに声をかけたり，体をふいてやったり，しつこくしないから，向こうから研究しに寄ってくるの」

ホンニャアはおばあちゃんが新聞などを読んでいると，近くにきて体を横たえたり，うす目をあけてタヌキ寝入りをしながら観察しているのだそうだ。

「サービスが過剰なのはめいわくなのよねぇ，トオル君」と，おばあちゃん。みんなめいわくしていると言いたいのだ。

たしかに，トオルのほうは呼ばれたら返事くらいはするけれど，ホンニャアは反応をしない。再三登場している猫の先生ピンキー君も，「呼ばれるたびに行かなくてもいい」と教えている。ホンニャアも空腹のときだけ，ちゃんとやってくる。自分にとって益があるかないか，猫のほうが相手をしっかり判断しているらしい。

「この猫は長生きね」と，またおばあちゃんが言った。「おばあちゃんもね」と言いそうになったけれど，名高い双子のおばあさん「きんさんぎんさん」に比べたらまだまだだ。

FBIのメンバー「ぶん」さんこと，テクニカルライターの河田文雄さんによると，栄養と医学に恵まれた日本の現状では，年齢の七掛けを実質的な年齢と考えるべきなのだそうだ。たしかにこれまでにつくられた年齢のイメージより若々しい人が，周囲にも多くなった。七掛けかあるいは八掛けというのも賛成できる。

100歳の「きんさんぎんさん」のように3ケタになると，八掛けくらいにするとして約80歳。あの頭のサエは，これまでの100歳とはちがう。おばあちゃんは77歳なので，0.75倍としても58歳だ。

さて，同じ環境で生活している日本の猫もこの計算法を適用できると考えれば，現在8歳のホンニャアをまず人間の年齢に換算して，56歳。猫は人間の7倍の成長をするそうだ。そして0.7倍すると39歳くらいということになる。ホンニャアもまだまだ長生きできるかもしれない。

連載を開始した1985年といえば，夫がパソコンをはじめてから10年目くらい。彼は化学会社のしごとのかたわら，よくパソコン誌に記事を書いていた。ときどきその記事にイラストを入れていた私のところに，当時の「Oh!MZ」の編集長Y氏がおとずれた。

「パソコンとの生活を書いてください。パソコンをあやつる人たちが野蛮な人種なのか，使えない人間が野蛮人なのか，さぐってほしいのです」

私たち家族と夕食をともにしながら，その日Y編集長はパソコンへの夢をたくさん語った。

「タイトルはどんなものになりますか」

会談がすんで椅子から立ち上がりかけたY編集長に私は聞いた。来客を警戒して，はじめからコソコソ逃げまわっていた白猫が，となりの部屋に走り去った。そのうしろ姿をチラと見ながらY氏は，

「『猫とコンピュータ』でいいでしょう」

とあっさり言った。あまりの即答に冗談かもしれないと思ったのだが，ほんとうにそのままになった。アナログとデジタルをズバリつなげたタイトルだ。

長期保存メール

連載のパートナーになったのが，編集部推薦のMZ-1500。私にとってははじめてのマシンで，夫にとってはポケコンも数えたら16台目のマシンだった。

そのころはどれがメインということもなく，書斎のラックにはほとんどのマシンが並べられていて，キーボードは片端からホンニャアの昼寝台になった。

MZ-1500は，ビギナーにもすぐに楽しさが感じられるよう，くふうされた機種だった。クイックディスクのシステムは画期的で，音楽，ゲーム，プログラムと，ひととおり取り揃えて，パソコンを動かす気分をもたせてくれた。マリオブラザースのMZ-1500版にも熱中したものだ。

数日前に，夫は「PCUA-BBS」という通信ネットにアクセスした。これは元の「JMCC-BBS」で，運営していた「日本マイコンクラブ」が，1991年に社団法人「パーソナルコンピュータユーザー利用技術協会」となったとき改称された。運営側の協会では，ソフトバンクの孫社長も理事をつとめておられる。

旧「JMCC」は1984年開局の，日本ではもっとも早い時期にできた通信ネットであり，構成メンバーもこの道の専門家が多いそうだ。日本マイコンクラブの会員だった夫も1985年に加入，８番目のIDを得ている。

おどろいたことに最後のアクセスは２年５カ月前だったようだ。1990年１月３日にシスオペから夫にあてられたメールが，未開封のままになっていた。それ以前の記録にもビックリ。夫が「86-01-26 17：29」に発した KIO-02075氏あてのメールが読まれたのは，「88-01-30 21：27」だそうだ。

通信は個々のペースでおこなわれるものだから，誰もがこんなふうではないだろうけれど，秒単位の記録を残しながら片道２年の超低速通信をしているところが，チグハグでユーモラスだ。

この由緒あるネットは，８年たったいまでも会員がようやく500名ほどで，電話回線もひとつ。あとからあらわれた規模を誇るネットのように華々しく膨張することもなく，記念碑的な存在のようにも見える。

「JMCC」開局の翌年，1985年の６月，パソコンクラブ「FORSIGHT」は，東京近県のある旅荘で研修の合宿をおこなっていた。音響カプラを使って公衆電話と接続し「JMCC」にアクセスをこころみたりしていたのだ。その１カ月あとにメンバーの一員である中村守利氏によって，テスト局としてスタートしたのが，「FBI-NET」だった。

いま関心の的になっているパソコン通信だが，この草分け的なBBSに入会を許されたことが，私にとってパソコンとのカルチャー摩擦を取り払う大きなきっかけとなった。

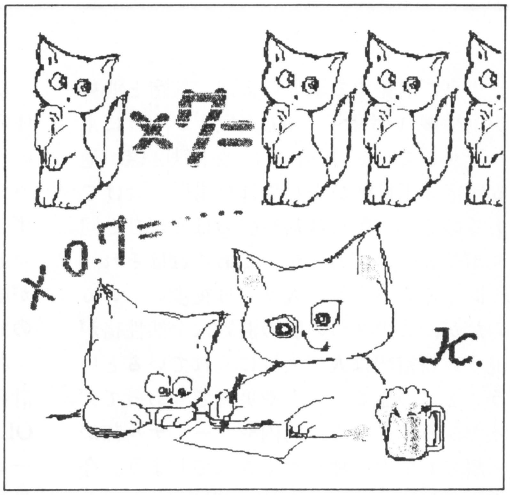
もっとやわらかく

通信をはじめたら，目の前からパソコンが消えた。パソコンの向こうにヒトがいることで，気づかぬうちにパソコンが機械としてではなく，自分の言葉や意思の一部であるような，しぜんなはたらきをするようになっていた。

これはコンピュータの一面にすぎないと知っても，新しい文明にふれた興奮でキーボードがグラグラになるまで叩き，コンピュータは，どこかの宣伝コピーのように，やわらかな，あったかいものになった。

タイピングもみるみる上達(？)し，みごとなブラインドタッチ。キーボードを見ては打たない。ただしローマ字変換。

パソ通では交遊の規模にも，作法にも新しい世界があって，いろいろな面で意識を改革させられたり，刺激を受けたりする。その点も大きな収穫となった。



illustration：Kyoko Takazawa

「40分じーっと待ってるの，動いたらカッコ悪いんだ。それで最後に28秒だけボクが叩くの」

「ほんとなの？　そりゃたいへんだ」

「そう，演奏より，動かないで待ってるほうがむずかしいの」

リビングで，トオルとおばあちゃんが話している。

都立Ｒ高校のオーケストラ部の卒業生だけで構成している「淡交フィルハーモニー」という交響楽団が，５月に恒例の演奏会を開いたときの話だ。60代の人もまじるアマチュアの楽団だが，プロの人もいる。

今回の指揮者は作曲家で編曲家の鈴木一行さんだった。卒業生のおひとりで，テレビの「題名のない音楽会」などでしごとをされているそうだ。

曲目はビゼーの組曲「アルルの女」で，現役２年生のトオルは，最後の曲，８曲目の最終部だけに使われる大太鼓のパートを担当した。

「28秒だけなら，誰かほかのパートの人が兼ねられなかったの？」とおばあちゃんが聞いている。

「ダメなの，フィナーレで全部のパートが演奏するから。生物の先生が交響楽団の団長なんで，たのまれたんだよ」

40小節，28秒の大太鼓の演奏は，現役のオーケストラ部員が引き受けるしかなかったようだ。しかも何日間も，出番のない長時間の練習に参加させられた。

「練習でボクが，ここぞとばかりに思いっきりひっぱたいたらね，リズムは正しいけれど，そういう叩きかたは響きに時間差ができてズレてしまうって，指揮者に注意されたの。それで，力まかせにではなく，セーブしながら叩くようにしたの」

「太鼓ひとつでもむずかしいもんだネ」

おばあちゃんは音楽通である。ほんとは先生になるより声楽家になりたかった。第１回のNHK全国のど自慢コンクールに，歌曲の部の代表として決勝戦までのぞんだ経験もある。

トオルは演奏会の録音テープを持ってきて，組曲の８曲目，最後の曲をおばあちゃんに聞かせた。誰でも知っているなじみのある旋律である。

最終部がきた。前の７曲をふくめてここまで40分，トオルが，まさしく満を持してのぞんだ価値ある28秒だ。フィナーレの大太鼓は，心なしか交響楽の量感を一気にもりあげ，荘重に響きわたった。

「これは値打ちがある。それによく引き受けてえらかったね」

孫にはいつもやさしいおばあちゃん。でもコンサートでは，演奏している人の個人的な事情は，できれば知らないほうがいいと思う。トオルはごくろうさまだったけれど，なんだかとっても笑ってしまいそうな話だから。

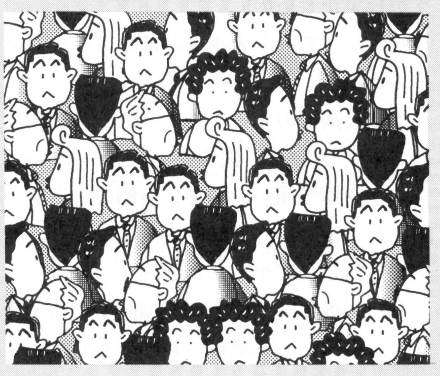
ホンニャアがいて，トオルがいて，おばあちゃんがおとずれるリビングは，外は雨でもあいかわらず明るい。

マシンルームのパソコンたちは長い間にずいぶん整理されて，メインのいくつかを残し押入に眠っている。パソコンと暮らしながら，パソコンがどこまでやわらかくなってくれるのか，もう少し見届けてみよう。

X-OVER・NIGHT
(クロスオーバーナイト)

[第25話]

IKEBUKURO



TAKAHARA HIDEKI 高原 秀己

東京で混雑する駅といえば，新宿と池袋が両横綱。いずれも300万人以上が１日に乗り降りしている。新宿には，地下鉄２線が通る新宿３丁目駅が少し離れた場所に設けてあるので，実際には新宿駅のほうがひと回りボリュームがあるが，まあそれはそれ。

同じくらい大量の人々が日夜歩いているのだが，この２つの駅の混雑は全然性格が違う。新宿駅は人が大量に流れているということにつきる。ストや災害でもあってダイヤが麻痺したら，構内やターミナル部が一瞬にして人で埋めつくされてしまう。今春の私鉄ストのときのJRの構内は，とにかくひどかった。地下コンコースは，人の生首がカーペットのように敷きつめられていたように見えてしまったほどだ。

しかし，これは特別なとき。通常時は，新宿駅では人の流れ方はおおむね一定している。導線に沿って，いくつかのパターンにはまった動き方をするので，大量に人はいるけれども，流れは非常にスムーズだ。

その点，池袋駅はまったく違う。最も他人とぶつかる駅といってもいい。メインとなる導線もなければ人の流れもない。混雑しているという次元ではなく，超大量の人がターミナル内に乱雑に充満しているとでもいおうか。狭い部屋の中に米と大豆と小豆を一度にブチまけたような乱雑さだ。

この理由だが，JR，営団地下鉄，西武，東武の４社が乗り入れており，しかも両私鉄と地下鉄とJRの一部が始発駅になっていることがまず指摘できる。始発駅ということは，別の路線から乗り換える人が大多数ということを意味している。

後背地が埼玉県であり，西東京や神奈川県をバックにしている新宿とは違うことも理由だ。といっても品格の話ではない。埼玉は人口増加率が全国ナンバー１。人口自体も東京，神奈川を猛追している。それでいて西東京や神奈川と違って，娯楽性の高い商業集積地域が埼玉県下には乏しいこともある。川崎，横浜はもちろん，吉祥寺，下北沢などに相当する街が少ない。だからどうしても通勤通学時に池袋に降りて，用事をすませようということになる。

この結果が，西武百貨店池袋店が年商4000億円で全国百貨店のトップ，という実績につながっているのである。ちなみに埼玉県は，政府のある調査で「最も豊かでない県」と指定されている。

だが池袋駅の乱雑さは，なによりも駅自体の設計の劣悪さと利用客層の２つが最大の理由であるとぼくは思う。各鉄道がバラバラに配置されていて，導線もはっきりせず迷路のようだ。だからただでさえウロウロとするのに加えて，改札口と切符売り場が離れている路線まである。こうなると人の流れがスムーズになるわけがない。

利用客層が多様すぎるのが，この駅の設計の悪さに輪をかけている。ビジネスマン，OL，学生から買い物客。(暴)関係の人やホームレスも多い。さらに日本人だけでなく外国人もかなりいる。とても老若男女などという４文字熟語ではいいつくせない。こうした人々が思い思いに道を間違え，切符売り場を探しながら歩いているのだから，池袋駅のターミナルが大混乱するのは，当然といえば当然なのだろう。

この池袋，西武百貨店やパルコ，サンシャインビルが集まるセゾン王国の東口はともかくとして，西口方面は上野や新橋と並ぶキタナい街だったのだが，東武デパートの拡張工事とJRの新駅ビル建設が終わって，ガラッと一変した。ひと言でいうと，有楽町マリオンよりもデラックスで豪華な雰囲気になってしまったのである。関係者ですら「やりすぎだ」というほど。

実際には「トイレの中は前のままでキタナい」とか「周りにはキタナい場所が結構残っている」とかいう声も出ているが，とにもかくにもあの“ダサイケブクロ“が変身したことは間違いない。東武デパートは，売り場面積が83000平方メートルで日本一になったし，中の売り場はすべてが化粧品か宝飾品売り場のように変身した。ピカピカ。よくも短期間であれだけ変えたものだと感心する。JRと東武の鉄道資本のパワーといえばそれまでだが，やる気ひとつでこれだけ変わるのかという好例だろう。

これで少しは池袋駅も歩きやすくなってくれればいいのだが，ますます混雑することは目に見えているし，実際に物見遊山の人が集まって，乱雑さに輪がかかっている。とりあえず，あちこちでゴロゴロとしていたホームレスのおじさんたちの姿が見えなくなったことはよし，としたい。

＊　＊　＊

今回はここの部分までまったくコンピュータの話題が出てこなかったので，関連づけて書くことはあきらめて，まったく唐突にセイコーエプソンの話に振ることにする。

なんでも，PC-9801互換機の売れ行きが落ちて儲からなくなってきたので互換機戦略を見直すという。低価格製品は極力減らして高額ビジネスパソコンへと主体をシフトするというのだ。なにか互換機ビジネスを勘違いしているのではないだろうか。

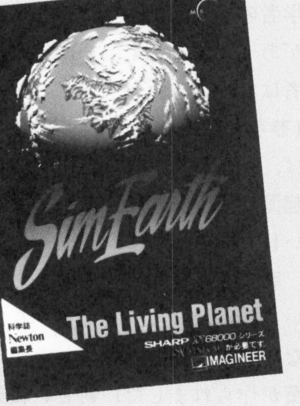
そもそも互換機などというのは，贋作とはいわないまでも，ある種の模倣品であることには間違いない。オリジナルメーカーに対して競争力が発揮できるのは価格だけ。その価格競争を投げ出してしまっては，なにも残らないことがわからないのだろうか？と疑問に感じてしまう。おそらく，性能で勝負だのシステム力で対抗だのと，僭越なことを考えているのだろう。しょせんは日本電気のアーキテクチャを追って作っただけの互換機だ。ハードだってインテルが苦労して開発したチップを買ってきてCPUに使うだけ。孫悟空が釈迦の手のひらから逃れられないように，この路線から逸脱した斬新な製品など作れるわけがない。

ただただ安くニセモノを作ることだけに企業使命をかけて専念してもらわなくてはならないのだ。常に定価で３割安の互換機を出せば，着実に売れる。すべてのNEC対抗機メーカーが失敗し続けた理由が「さして変わらんものをNECより高く売るからだ」ということを，どうしてどこも理解できないのか，ぼくにはわからない。

愛読者プレゼント

イマジニア　☎03(3343)8911

1 シムアース



X68000 5"2HD版

12,800円(税別)　2名

ガイア理論に基づいて作られた，惑星成長過程シミュレーションゲーム。初のSX-WINDOW用パッケージゲームとあって，移植には結構時間がかかったが，ようやく発売された。「よっ，待ってました！」。

アルシスソフトウェア　☎0956(22)3881

2 スピンディジーII

X68000用 3.5/5"2HD版2枚組

8,700円(税別)　3名

コマを操って，パズルのような面をクリアしていく。右に行ったり，左に行ったり，ジャンプしたり，ボタンを踏んだりしながら，謎を解かなければ先に進めないのだ。

ソフトバンク　☎03(5488)1360

3 X68000マシン語プログラミング グラフィック編

3,600円(税込)　5名

"入門編"に続く，「X68000マシン語プログラミング」の第2弾が，昨年発売されたこの"グラフィック編"。遅ればせながらも5名の皆様に。

4 飲み物他詰め合わせ

1名

各方面から寄せられた，各種変わりダネをかき集めて1名の方にプレゼント。どれを誰にもらったかは，もはや定かではないが，毒は入っていないと思われるので，ご安心を。

プレゼントの応募方法

とじ込みのアンケートはがきの該当項目をすべてご記入のうえ，希望するプレゼント番号をはがき右下のスペースにひとつ記入してお申し込みください。締め切りは1992年8月18日の到着分までとします。当選者の発表は1992年10月号で行います。

6月号プレゼント当選者

1 Aレミングス　(栃木県)柿沢雄大　(東京都)大山和紀　(神奈川県)木村英正　八木明　(静岡県)佐藤庄一　(愛知県)鈴木康夫　(滋賀県)森夕香　(京都府)富井雅男　(広島県)大谷洋徳　(山口県)鈴木正紀　Bシムアース　(福島県)伊藤直広　(東京都)西村直也　渡辺圭一　山口隆久　箕浦真　(群馬県)野原慎司　(山梨県)石井一成　(神奈川県)海津雄一　(兵庫県)多田哲也　(広島県)堀岡俊秀　2 スターモビール　(栃木県)神生直敏　(神奈川県)岩下尚人　堂領輝昌　(静岡県)小倉健　(兵庫県)今田智宣　3 AジーザスII　(宮城県)及川清孝　(新潟県)金子聡　BワールドゴルフIII　(岡山県)山下善之　(大分県)河野真司　Cドラクエ下敷き　(香川県)長谷川聖 他4名　Dソウルブレイダーテレホンカード　(群馬県)竹内正人 他9名　4 Aナイトアームズ　(北海道)小林勝　(新潟県)竹之内貴広　(千葉県)伴武士　BスピンディジーIIボールペン　(鳥取県)山中明博 他9名　5 エトワールプリンセス　(東京都)富田祐樹　(神奈川県)澤田裕史　(兵庫県)林田誠司　6 ピンボールピンボール　(神奈川県)姉帯寛　(愛知県)金田充弘　(岡山県)中野孝行　7 Aライフ＆デス　(大阪府)森本賢　B超人　(大阪府)今井彰彦　Cヘビーノヴァ　(北海道)荘司真吾　8 グラディウスII　(静岡県)池田健一　9 Aゲーメストオリジナルタオル　(群馬県)久保田智久 他4名　B同ボールペン＆シャープペン　(新潟県)山中雅彦 他9名　C B-TYPEオリジナルテレホンカード　(愛知県)鵜野亘 他9名　Dゲーメストオリジナルストリートファイター IIノート　(兵庫県)春名義行 他19名　E出たな!! ツインビーノート　(愛媛県)越智浩之 他9名　10 カプコンF3エントリー記念テレホンカード　(三重県)朝倉龍一 他9名　11 SPSオリジナルテレホンカード　(愛知県)笹山和宏 他4名　12 A XENON2　(栃木県)木村直也　(大阪府)石原英治　Bドラッケン　(宮城県)船山正和　(千葉県)清水博　13 AファーストクィーンII　(北海道)下広崇英　(東京都)椎名美臣　(兵庫県)櫟利春　B同ポスター　(千葉県)横須賀稔 他9名　C同オリジナルCD　(東京都)黒崎亨　(長野県)須澤加実　(愛知県)奥田直也　14 銀河英雄伝説II　(大阪府)仲村朋睦　(兵庫県)林秀彦　15 工画堂スタジオオリジナルボールペン　(大阪府)北野明 他9名　16 Aジェミニウイング　(山口県)遠藤耕二　Bシステムサコムオリジナルブルゾン　(愛知県)安川実　(愛媛県)山内正哉　17 A F15ストライクイーグルII＆シナリオ　(静岡県)平山大介　(兵庫県)井上卓顕　Bガンシップ　(和歌山県)加納伸吾　斉藤勝文　Cマイクロプローズジャパンオリジナルステプラー　(三重県)橋爪英記 他5名　18 JOYCONTターボIV　(神奈川県)森河良太 他10名

(敬称略)

以上の方々が当選しました。おめでとうございます。そして，Z'sSTAFF PRO-68K ver.3.0モニタ募集の当選者は岐阜県の今村雄治さんに決定しました。商品は順次発送いたしますが，入荷状況などにより遅れる場合もあります。雑誌公正競争規約の定めにより，当選された方はこの号の他の懸賞には当選できない場合がありますのでご了承ください。

なぜ13分で料理が消えたのか

学会での小さな事件

6月15日から17日まで，横浜の「みなとみらい」というずいぶんと綺麗な人工都市の一角の，これまた綺麗なホテルで，総計300人にも及ぶという大きなシンポジウム，つまり学会（の催し）があり，僕も参加してきました。1日目は昼から始まり夜まで，2日目は朝から夜まで，3日目は朝から昼までという日程で，発表は3つのパラレルセッションでした。

ネットワークとか言語処理系とかいうように，同じ分野の発表が3本ずつまとめられてひとつのセッションとなり，それらは連続して発表されます。ひとつの発表は質疑応答も含めて30分です。パラレルセッションというのは，そのようなセッションが複数同時に開催されることを意味します。ですから，聞く人は3つのセッションのうち，好きなテーマに関して発表されているセッション会場に行きます。望むならば，30分ごとに別の発表を聞くために会場を移動してもいいのです。

学会が開催するこのようなシンポジウムや会議には必ずといっていいほど「懇親会」が組み込まれており，参加者たちが飲み食いしながら歓談する光景が見られます。こういうシンポジウムの意味は懇親会にこそあるのだと思っている人も少なくないかもしれません。別に，その人が飲んべえだとか大食いだからとかいうわけではありません。こういうくだけた場で，初対面の人と面識ができたり，情報交換したりできるからというのです。

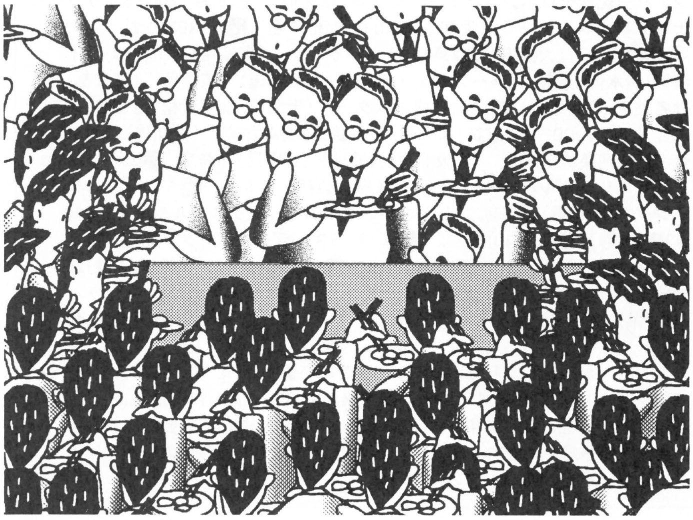
これは，確かにもっともな意見といえましょう。発表そのもののほうは，一種の自己アピールの場であり，また時間も限定されていますから，その研究をよく理解したり，それを聞いて自分の研究を深めたりするにも，それなりの限度があるといえるからです。

このシンポジウムでも，例にもれず，1日目の夜の6時から8時までの時間に懇親会が設定されていました。もちろん，僕も出席しました。懇親会の会場は，発表のための3つのセッション会場のすぐ隣にありました。シンポジウム会場は，海に面したところにあるたいへん綺麗な建物の中にあり，懇親会場の雰囲気もいいものでした。

さて，予定どおり6時から始まった懇親会場から，開始後13分でこつぜんと姿を消してしまったものがあります。それが問題の料理です。姿を消した料理はその後1時間47分のあいだ，決して戻ってきはしませんでした。豪華な懇親会場と空の皿たち，虚しいものでした。

神の英知を愛していた研究者

懇親会場にいた200人以上の人々。それはいったい，いかなる人たちなのでしょう？



ひと言でいえば研究者たちです。もう少し正確にいうと，それはプロの研究者，研究することによってお金をもらって生活している人です。唐突な質問かもしれませんが，彼らのまわりには何か神聖なる雰囲気が漂っていたでしょうか？　もちろん，その答えはNOです。しかし，料理が13分で姿を消したからというのではなく，そもそも本来その答えはNOということなのかもしれません。

今日の科学者の本性を鋭くエグろうとしている文章があります。村上陽一郎氏が書いた「研究者はいつから愛を失ってしまったのか？」（参考文献）という一文です。ここで村上氏は，現在の「プロの」研究者に対してその起源や遍歴をたどりながら，きわめて厳しい目を向けています。

そもそも，職業としての研究者，あるいは科学者というのは，たかだか150年ぐらいしか歴史のない（1840年頃サイエンティストという言葉が作られました）新しい職業なのだそうです。それ以前は，研究する人々のことを「フィロソファ」と呼んでいました。

現在は，この言葉は哲学者という狭い意味で使われますが，昔は，自然科学，人文科学を問わず，知識を追究する人につけられていたのです。フィロソファ＝フィロ（愛する）＋ソフィア（知識）という語源のとおり，純粋に知識を愛する人という意味であり，これを職業にするなどという考えは一切存在しなかったのでした。

それどころか，フィロソファはキリスト教的意味において，きわめて崇高で貴い人々であったのです。なぜならば，神の書いた第1の書が聖書で，第2の書が自然ということであり，知を追究すること，つまり自然についての知識を追究することは，神の作品を愛すること，神の英知を追究することにほかならないことだからです。知識を切り売りして金銭を得るなどということは，神に対する冒瀆なのです。これが17世紀頃までの話です。

研究者は愛を失ったのか？

18世紀になって起こった啓蒙運動によって大きな変化が起こりました。啓蒙主義者たちは，知識というものをキリスト教から解放しようとしたのです。そのような運動によって生まれたのが，「バラバラにされた知識」としての百科事典というわけです。

知識を愛するということをしなくなった人々を駆り立てたもの，それは，上手にパンを焼いたり，効率よく石炭を掘り出すといった，具体的に自分の力を高めようとする意志でした。

19世紀になると，百科事典のようなバラバラの知識は不都合であるということで，再編成が必要ということになってきました。再編成するという作業は，同時に各分野において，その部門だけを専門的に研究する専門家の登場を促します。バラバラの知識を再編成するときに，ここは自分の領域だと主張する人々が生まれたのです。その人たちこそがサイエンティストというわけなのです。

19世紀の大学改革によって，高度な知識をもつ人間を作り出す機関としての大学が登場し，これにより，ついにアマのフィロソファとは本質的に異なるプロのサイエンティストが誕生してきたのです。このことは同時に，知を愛すること，あるいは，神に対するような敬虔さを必要とされないようなプロの研究者の誕生をも意味するのです。

村上氏は，さらに19世紀後半から現在に至る研究者の姿に対して，容赦ない批判を投げかけます（現在の欧米の大企業は，もともとは個人の発明家にその起源をもつものが多いのですが)。発明家たちが企業で莫大な富を築くのにしたがい，研究者たちも，大学だけでなく私企業にだんだんと引きずり込まれていくようになります。

知に対する愛などは必要のない研究者像の生産にひと役買っているのが，かつては神の英知を知るための特殊機関であった大学であると，村上氏は指摘します。教師は，教育という行為によって，どういうふうに書けば論文が受諾されるか，どういうふうにすれば研究者の仲間として認められるかということを学生に叩き込み，自分のコピーを作っているというのです。

さらに，村上氏は，ノーベル賞に象徴されるような仕掛けによって育成された，まわりを蹴落すことによって這いずり上がるといった風土の中で，研究者に対して倫理に関する問題などを要求するのが，そもそも無理なのだと書いています。

謎の答えは風の中に

研究者の意識うんぬんがどうしたこうしたなんて，そもそも大袈裟であって，単に出された料理の分量が少なかっただけなのかもしれません。しかし，そう簡単に片づけてよい問題でもないのかもしれないのです。

証言その1

「料理の打ち合わせのときに，なるべく質より量で，と頼んでおいたんですがねぇー」

証言その2

「給仕の人が，この人たちは皆さんお昼を抜いてらっしゃったのですかねぇ，と皮肉をいってましたよ」

僕自身も懇談会が始まる前に，消え去る前の料理を確認しましたが，素晴らしそうな料理が確かに各種並んでいました。そして乾杯が終わり，僕の立っているところ(立食パーティ）の近くに，久しぶりに会う後輩がいたので数分立ち話をし，「さて，料理を」と取りにいくと，すごい人だかりで，やっとのことで小さめのお皿に2/3程度，何種類かの料理を盛って戻ってきました。そして，また別の人と話をしていると，パーッと人が料理のまわりから引くような気配がしたので，そちらのほうをよく見てみると，料理がすっかりと跡形もなく消えていたのです。

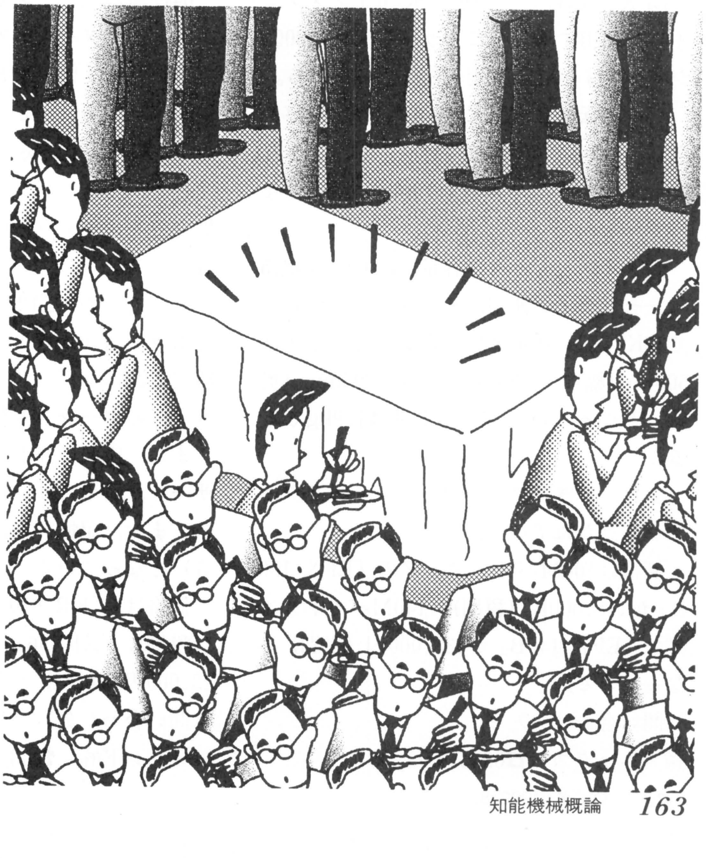
あとから聞いた話では，幹事の先生はあわててホテル側に料理を追加注文したそうです（それはあせったでしょう，ご心中お察し申し上げます)。確かに1時間ぐらいして数皿の追加分の料理がきたようで，ほんのちょっとのあいだだけ小さな人だかりができたのを記憶していますが，いま考えれば，それは文字どおり焼け石に水でした。

僕は，この小さな小さな事件は，この学会のもつ自由で気さくで格式ばらない，いい雰囲気を象徴しているのであろうと思います。先に紹介した村上陽一郎氏の文章におけるような，「研究者からのフィロソファ的性格の喪失」を微塵も表していないでしょう。

どうせ，たぶん，おいしそうな刺し身(確かあったと思うが）や高級料理を取り逃がして，隅っこにあった料理をほんのちょっとしか食べられなかったことを，単に恨んでいるだけなのでしょう，チャンチャンッ。

参考文献

村上陽一郎：「研究者はいつから愛を失ってしまったのか？」，研究する人生，260-271ページ，JICC出版局（1991).

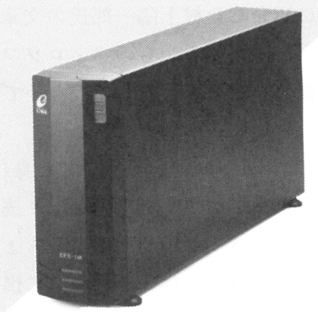


PENGUIN INFORMATION CORNER

ペ・ン・ギ・ン・情・報・コ・ー・ナ・ー

NEW PRODUCTS

X68000用ハードディスク
EFX-100B/140B
エニックス



EFX-140B

エニックスは7月中旬よりX68000用ハードディスクを発売する。

今回発売するのは，100Mバイトタイプの「EFX-100B」と140Mバイトタイプの「EFX-140B」の2機種。SCSIインタフェイスを使用することにより，従来のX68000シリーズすべてに使用可能となっている。

データ転送は「EFX-100B」が1.5Mバイト/s，「EFX-140B」が4Mバイト/sであり，シークタイムはそれぞれ19ms，16msとなっている。

両タイプともセクタ容量512バイトの3.5インチドライブ使用，データバッファとして32Kバイトを確保している。外形寸法は，50mm（幅）×125mm（高さ）×295mm（奥行）となっていて，本体重量は1.9kgである。

また，本機はブラックとグレーの2タイプが用意されている。

価格は100Mバイトタイプの「EFX-100B」が118,000円（税別），140Mバイトタイプの「EFX-140B」が138,000円（税別）となっている。

〈問い合わせ先〉
㈱エニックス　☎03(3367)1908

ローランドピアノ・デジタル
KR-650
ローランド

KR-650

ローランドは，デジタルピアノ「KR-650」を発売した。

本機は，上位機種「KR-4500」の機能をほぼ継承した。GENERAL MIDI対応の76鍵デジタルピアノである。

64種類のパターンを持つ自動伴奏機能と，ピアノ・スタイル・アレンジャー機能で鍵盤を演奏用，伴奏用と区別せずに自由な演奏を自動伴奏付きで楽しめるようになっている。これらの演奏データは，内蔵の3.5チインチディスクドライブに保存でき，また，別売のミュージックデータを利用することもできる。

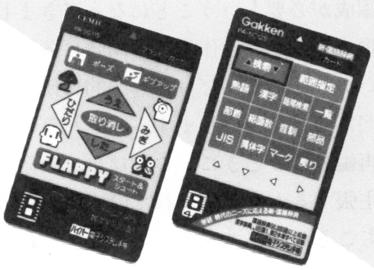
そして，標準で128音色，ドラムセットを8セット，効果音1セットを搭載。最大同時発音数は28音，8タイプのリバーブ，4タイプのコーラスも装備している。

また，内蔵の6トラックソングコンポーザーで，簡単に多重録音をすることができる。まお，記憶容量は5ソング（約18,000音）となっている。

価格は，「KR-650（本体）」が298,000円（税別），「KRS-650（専用スタンド）」が23,000円（税別）となっている。

〈問い合わせ先〉
ローランド㈱　☎03(3251)5595

電子システム手帳ICカード
PA-5C11S/12S
シャープ



シャープは新しく電子システム手帳用のICカード2種類を発売した。

○フラッピーカード「PA-5C11S」（8行表示専用カード）

パソコンゲームでお馴染みのアクションパズルゲーム「フラッピー」が，ハイパー電子システム手帳対応カードとして登場した。

ゲームはパソコン版と同じく，フラッピーを操りブルーストーンを指定の場所へ戻していくというルールである。ステージ数は全100面あり，コンティニュー機能や5面ごとに現れるパスワードによって，好きなステージからゲームを遊ぶことができる。

価格は6,000円（税別）となっている。

〈問い合わせ先〉
㈱セミック　☎03(3582)6821

○新・国語辞典カード「PA-5C12S」（8,4桁表示用カード）

このカードには約41,000語の見出し語，約9,000語のカタカナ語・略語，6,355文字の漢字（JIS第1水準，第2水準），524種類の記号が収録されている。

見出し語検索，熟語検索，語尾検索機能を備え，意味を調べたうえでさらにわからない言葉を指定して検索することも可能となっている。漢字には音訓，総画数，部首による検索条件を組み合わせることもできる。さらに細かく，その漢字を構成してい

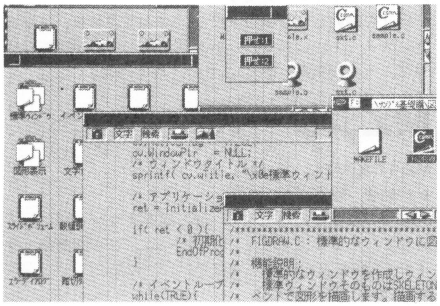
る文字を手掛かりに検索する部品検索機能もある。

価格は18,000円（税別）となっている。

〈問い合わせ先〉

㈱学習研究社　☎03(3726)8518

SX-WINDOW開発ツールキット
CZ-288LWD
シャープ



シャープは，SX-WINDOW上でソフト開発を行うための開発支援ツール「CZ-288LWD」を発売する。

開発ツールとして，複数のプログラムを同時にデバッグ可能なソースコードデバッガ「SXデバッガ」，リソースを作成・編集可能な「リソースエディタ」，リソースデータをリンクしてリソースファイルを作成する「リソースリンカ」，XC付属のMAKE.XをSX-WINDOW上から利用する「サンプルメイク」を装備。そして，初めてウィンドウプログラムを組もうとしている人のために，基本機能を理解するためのサンプルプログラム（33種）も用意されている。

なお，本ソフトの使用には，メインメモリ4Mバイト，SX-WINDOW ver.2.0以上，C compilerPRO-68K ver.2.0が必要となっている。価格は未定。

〈問い合わせ先〉

シャープ㈱　☎03(3260)1161，06(621)1221

ファックスモデム
EPSON MX-240
セイコーエプソン

セイコーエプソンでは，ファクシミリ送信機能と全2重データモデム機能を一体化させた「EPSON MX-240」を発売した。

ファックスモデムモードでは，最大9600bpsでCCITT T.4/T.30に準拠するG3ファクシミリへの送/受信機能を搭載。ファクシミリ通信規格としてEIA-578規格（通称クラス1）を採用している。データモデムモ

EPSON MX-240

ードでは，セルラー通信（自動車電話など無線を媒体とする通信機能）に対応するMNPクラス10を搭載。エラー訂正機能としてV.42bisおよびMNPクラス2～5をサポートしている。

サイズは，114mm(幅)×30mm(高さ)×174mm(奥行)で，重量は370g。インタフェイスはRS-232Cを使用している。

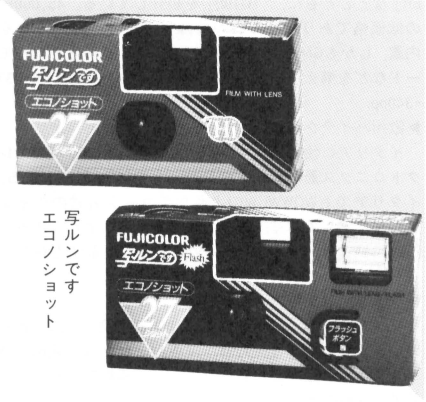
価格は44,800円（税別）となっている。

〈問い合わせ先〉

㈱エプソンインフォメーションセンター

☎03(3377)3500，06(399)1115，052(953)9239

効率アップ，コンパクト仕様
写ルンです エコノショット
富士写真フイルム



写ルンです エコノショット

富士写真フイルムは従来の「写ルンです」を小型・軽量化した「写ルンです エコノショット」シリーズを発売した。

この「エコノショット」では小型・軽量化だけでなく，フィルムの予備スペースを使い，24/36枚撮フィルムでそれぞれ27/39枚の撮影が可能となっている。

価格は「写ルンですHiエコノショット27」が1,000円，「写ルンですHiエコノショット39」が1,200円，「写ルンですFlashエコノショット27」が1,800円，「写ルンですFlashエコノショット39」が2,000円（すべて税別）となっている。

〈問い合わせ先〉

富士写真フイルム㈱　☎03(3406)2981

INFORMATION

GAME MUSIC FESTIVAL'92

今年で3回目になるゲームミュージックコンサート「GAME MUSIC FESTIVAL'92」が，8月に開催される。

開催日は8月22日（土），23日（日）の2日間（開場16：30，開演17：00）で，場所は日本青年館となっている。出演者は，22日が「J.D.K.BAND」，「GAMADELIC」，「ZUNTATA」，23日が「ALFHLYRA」，「コナミ矩形波倶楽部」，「S.S.T.BAND」を予定している。

入場料は1日券が3,500円（税込），2日券が6,000円（税込）。チケットは以下の箇所で販売している。

グローバルエンタープライズ	☎03(3496)1341
チケットぴあ	☎03(5237)9999
チケットセゾン	☎03(5990)9999
CNプレイガイド	☎03(3257)9999
丸井チケットぴあ	☎03(3363)9999

〈問い合わせ先〉

㈱グローバルエンタープライズ ☎03(3496)1341

電話版ゲーム雑誌
TVゲーム情報ダイヤル
東京電化

東京電化では，ダイヤルQ^2サービスを使った電話版ゲーム雑誌「TVゲーム情報ダイヤル」を開始した。

ターゲットマシンは，ファミコン（スーパーファミコンを含む），メガドライブ，PCエンジン，パソコン全機種の4種類。ユーザーは，それぞれのターゲットマシンの情報を提供したり，交換したりすることができる。サービス地域（システム設地場所），アクセス番号は以下のとおり。

東京・関東地区（埼玉県朝霞市）
☎0990(300)448

関西・中部地区（愛知県四日市市）
☎0990(343)258

中国・四国地区（広島県広島市）
☎0990(337)305

使用料は10円/20秒（通話料別）。

〈問い合わせ先〉

東京電化㈱　☎0484(87)4391

FILES Oh!X

このインデックスは，タイトル，注記――著者名，誌名，月号，ページで構成されています。さあ夏休み。プログラミングに精を出すのもいいけど，たまには海や山に遊びにいきましょう。体力もしっかりつけなくっちゃね。

参考文献

I/O　工学社
ASCII　アスキー
コンプティーク　角川書店
THE COMPUTER　ソフトバンク
テクノポリス　徳間書店
POPCOM　小学館
マイコン　電波新聞社
マイコンBASIC Magazine　電波新聞社
LOGIN　アスキー

一般

▶HOT! INFORMATION
シャープのカラーインクジェットプリンタ「IO-735XW」を紹介。日本語MS-Windows3.0に対応し，B4横用紙幅の印字を実現。――編集部，マイコンBASIC Magazine，7月号，98p.

▶THE NEWS FILE
シャープとアップルが電子グッズで提携。シャープは自社の液晶・デバイス技術をアップル社に提供，アップルはコンピュータ開発技術とシステムソフトウェア技術をシャープに供与している。――編集部，LOGIN，11号，34p.

▶2002未来コンピュータ
10年後のパソコンの姿を考える特集。アスキーの提案する仮想マシンを提示し，鹿野司氏が未来のパーソナルコンピューティングを考察する。そのほか各専門分野でパソコンを活用する人へのインタビューや，坂村健と西和彦の対談なども掲載。――編集部ほか，ASCII，7月号，226-260pp.

▶CD-ROMって，こんなに便利
電子ブックのビュアーに新製品が加わった。Panasonicの「KX-EBP1」は心臓部に16ビットCPUを搭載し，従来機と一線を画すスピードを手に入れている。そのテストレビューと，電子ブックをめぐるトピックスをお届けする。――志村拓，ASCII，7月号，301-308pp.

▶Digi-Ana Valley
CDよりLPのほうが音がいいという一部のマニアの話は本当だった？　パイオニアから発売されたCDプレイヤー「PD-T09」は，本来収録されていない20kHz以上の高音を合成し，聞き疲れのしない音を作り出す。その試聴体験記と，「シートの色で音が変わる」という話を紹介する。――編集部，ASCII，7月号，309-316pp.

▶MiCRO MUSiQUES
ヤマハがいよいよデスクトップミュージック市場に参入してきた。ここでは「CBX-101/201」「MUSIC COMPOSER」などとともに，「TG100」を紹介している。45,000円の低価格でありながらAWM音源と8種類のリバーブを内蔵，しかもGMモードやCM-64のエミュレートを行うモードなどを備えている。――編集部，ASCII，7月号，337-340pp.

▶欧州ハイテク事情
イタリアに住む筆者がレポートするヨーロッパのエレクトロニクス最新事情。今回はペイTVをリポートする。イタリアでもWOWWOWのようなデコーダ方式のチャンネルが登場しており，最新の映画を見ることができる。これに関連してイタリアのTVチャンネル事情も報告。――菊池薫，ASCII，7月号，396-397pp.

▶フタを開けたらみんなパソ通していた
ASCII誌のコラムニスト7人による座談会。1人ひとりのいままでのパソコン経歴，環境，これからの期待とパソコン通信の話などが語られる。――編集部，ASCII，7月号，398-403pp.

▶いまが旬のシミュレーション・ゲームはこれだ!?
ソフマップ6号店，ツクモ本店＆7号店，ラオックスゲーム館の店員さんにシミュレーションゲームの売れ筋作品と傾向を聞く。――編集部，マイコン，7月号，93-96pp.

▶ゲームをより快適に
ゲームのプレイをいっそう楽しいものにしてくれる小物類が大集合。ジョイスティック，マウス，ビデオコンバータなどが取り上げられている。――武蔵光太郎，マイコン，7月号，106-108pp.

▶マイクロコンピュータショウ'92
5月22日から25日まで，東京流通センターで行われたマイコンショウの模様を伝える。ICカードメモリ，低電圧チップなど，技術力をアピールする展示が多かった模様。――仲みゆき，マイコン，7月号，117-120pp.

▶ビジネスショウ'92 TOKYO
5月20日から23日にかけて東京晴海の国際見本市会場でビジネスショウが開催された。マイコンショウとは異なり，コンピュータ製品中心の展示となったこのショウの傾向と出品作をレポートする。――山田憲一，マイコン，7月号，121-126pp.

▶見本市会場膝栗毛
見本市レポート。今回はTOTOのフェア，水について考えるAQUA-HUMANIA'92。健康管理をしてくれるトイレやペットのためのトイレなどが展示されていた。ほかに6，7月の見本市スケジュールも掲載。――編集部，マイコン，7月号，170-172pp.

▶光磁気ディスクMO-3120徹底使用レポート
ハードディスクの次の選択肢として現実味をおびてきている光磁気ディスク。アイシーエムの3.5インチ光磁気ディスクユニット「MO-3120」を取り上げて，使い方やパフォーマンスについて述べる。――高橋雄一，マイコン，7月号，329-338pp.

▶ブロッホライン・メモリ
日立の開発した新磁性メモリについて述べる。1平方センチメートルに10Gビットを詰め込むことが可能な代物だ。――編集部，I/O，7月号，150-153pp.

MZシリーズ

MZ-1500(BASIC 5Z-001)

▶移植版　香港
4つの条件にしたがって，同じ牌を2つひと組にして裏返す，上海によく似たパズルゲーム。――深田郎，マイコンBASIC Magazine，7月号，122-123pp.

MZ-2500(BASIC-M25)

▶塀　～問仲辛羊の場合～
学寮の門限に遅れてしまった問仲辛羊くん。門の前には厳しい寮母さん。成績優秀な彼は彼女の目を盗み，学寮に侵入することを企てた！　パスワード当てアクションゲーム。――アダモ，マイコンBASIC Magazine，7月号，124-126pp.

X1/turbo/Z

X1シリーズ

▶THE CLEANER
故障中の自機を駆って敵の攻撃をよけ，機動要塞をレーザーで破壊しろ。お手軽ワンキーシューティングゲーム。――高橋大吾，マイコンBASIC Magazine，7月号，149-150pp.

▶がんばれ編集長
上から落ちてくる雑誌や人の顔などを同じ種類が重なるように積み上げて消す。ベーマガのキャラクターを使用したアクションパズルゲーム。――中尾明，マイコンBASIC Magazine，7月号，151-153pp.

X1＋FM音源ボード（要NEW FM音源ドライバ）

▶サンダークロスⅡ Air Battle＝THUNDER CROSS Ⅱ
コナミのゲームミュージックプログラム。――たつくん，マイコンBASIC Magazine，7月号，171-172pp.

X68000

▶SOFT EXPRESS
ファンタジックな王女様の怪物退治「エトワールプリンセス」を紹介。機種別NEW SOFTインデックスなど。――編集部，コンプティーク，7月号，69・72pp.

▶新作の真相
格闘アクションゲーム「ファイナルファイト」の情報をいち早く紹介。――編集部，コンプティーク別冊付録，7月号，22-23pp.

▶GAMING WORLD
渋いけど新しい神業シミュレーション「ポピュラスⅡ」の第1報。前作と比較。アーケードゲームの人気作品から移植，格闘ゲームの草分け的存在である「ファイナルファイト」を紹介。――編集部，テクノポリス，7月号，22-27pp.

▶アルゴリズムを見切ったぞ!?
3次元ゲーム用座標変換の基礎講座その2。3Dゲームで必要な回転変換などをX-BASICのリストなどを掲載して解説している。――おにおん，テクノポリス，7月号，138-143pp.

▶Software Hot Press
第1回全日本X68000芸術祭のグランプリ作品「TORN

ADO」「ペンギンランドネット」を紹介。ゲームのほうは「ポピュラスⅡ」「ファイナルファイト」「バトルテック 〜奪われた聖杯〜」「ノア」「エトワールプリンセス」などを紹介している。――編集部，POPCOM，7月号，6-37pp.

▶X68000新聞
「エトワールプリンセス」「OVERTAKE」「棋太平68K」「MIC68K」などを紹介。――編集部，LOGIN，11号，302-305pp.

▶SUPER NEW SOFT
他機種に先駆けて発売が予定されている「ポピュラスⅡ」，開発途中の画面でその全貌を想像しよう。――編集部，LOGIN，12号，12-13pp.

▶X68000新聞
期待の一作「ファイナルファイト」は，原作ファンも満足の完璧な移植だそうだ。今回はゲーム以外のものが多く「Z'sSTAFF PRO-68K ver.3.0」「CHART PRO-68K」などの新製品の紹介ほか，バージョンアップされた「SX-WINDOW ver.2.0」のお知らせなど。――編集部，LOGIN，12月号，224-227pp.

▶足
足がアニメーションする。走り幅跳びのゲーム。――渋谷正徳，マイコンBASIC Magazine，7月号，154-155pp.

▶あみだばにっく
あみだくじを登ってくる敵を，おむすびを投下してやっつける。おむすびもあみだくじ式に落ちていくぞ。ランダムで作成したあみだくじを使ったアクションゲーム。――千吉良賢一，マイコンBASIC Magazine，7月号，156-157pp.

▶SPACE GANGS
迫りくる敵をショットで跳ね飛ばし，近づけないようにしよう。タイマーが切れるまで君は耐えられるか？コスモギャングズ風のシューティングアクション。――長谷川光助，マイコンBASIC Magazine，7月号，158-159pp.

▶FINAL FANTASY Ⅳ 〜黒チョコボのテーマ〜
スクウェアのゲームミュージックプログラム。要NAGDRV。――西丸由貴，マイコンBASIC Magazine，7月号，173-174pp.

▶フェリオス 〜勇者の碑〜
ナムコのミュージックプログラム。要NAGDRV＋CM-64系音源モジュール。――牧田竜也，マイコンBASIC Magazine，7月号，175-176pp.

▶誌上公開質問状
本体同梱のワープロでハガキに印刷をするには？ カラーディスプレイ「CZ-613D」のオーディオ入出力端子は左右別にあるか？ などの質問に回答している。――編集部，マイコンBASIC Magazine，7月号，177p.

▶AV STRASSE
アクセスとハドソンの共同開発によるV70CPUボード「VDTK-X68K」の製品概要と使用感をレポート。また，X68000最強のグラフィックツールとして「Z's STAFF PRO-68K ver.3.0」を試用する。――編集部，ASCII，7月号，333-336pp.

▶FREE SOFTWARE INDEX
ここ1〜2カ月の間に主要ネットにアップロードされたソフトウェアから，目についたものを選んで紹介する。X68000用にはバックグラウンドプロセス強化ソフトなどが挙げられている。――編集部，ASCII，7月号，411-417pp.

▶「眠れぬ夜」のシミュレーションゲーム
シムシティー以後，傾向を大きく変えたシミュレーションゲーム界。その最新の作品を取り上げ，そこに潜む面白さを探る。――猪野清秀，マイコン，7月号，77-92pp.

▶なんでもQ＆A
X68000 CompactXVIのSCSIインタフェイスとX68000用増設RAMの種類について，またSX-WINDOW上の使い勝手に関して答える。――シャープ株式会社AVCシステム事業推進室，マイコン，7月号，360-361pp.

▶国内・海外ニュース
シャープはX68000シリーズの1992年度販売目標を5万台に設定した。前年比20％増の販売を目指すとのこと。また，日本語ワープロと32ビットパソコンを一体化した「書院パソコン」を発売のニュースなど。――編集部，マイコン，7月号，372-376pp.

▶SLG Laboratory
光栄の歴史シミュレーションゲームの最先端「三國志Ⅲ」を取り上げ，漢中の小勢力である張魯を選んで誌上でリプレイする。――猪野清秀，マイコン，7月号，398-401pp.

▶X68000をビジネス仕様に変身させる
GUIや大容量メモリなどでMacintoshやDOS/Vと同様の特徴を意識しているX68000を，ビジネスマシンとしてとらえシステム構築してみようという企画。資質は十分だがソフト環境の整備が課題か。――田中利昭＋荻窪圭，THE COMPUTER，7月号，78-87pp.

▶LOOP
縦スクロール型のシューティングゲーム。ミサイルとレーザーで敵を倒し，最後にはデカキャラと対決するというお約束のつくり。――西田浩一，I/O，7月号，104-105pp.

ポケコン

PC-E500

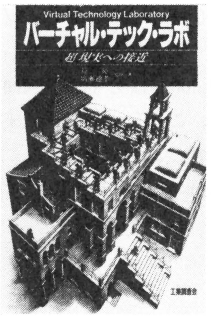
▶CORORON
コラムス風アクションパズルゲーム。――近藤紀之，マイコンBASIC Magazine，7月号，161-162pp.

新刊書案内

グリフォンズ・ガーデン
早瀬耕著
早川書房刊
☎03(3252)3111
新書判 286ページ
1,700円(税込)

一橋大学の金子郁容教授のゼミでは，卒論はなんでもいいらしい。でもって，卒論として小説を提出した学生がいたらしい。その卒論が，この「グリフォンズ・ガーデン」である。小説が卒論，っていっても，一橋大学に文学部はない。商学部経営学科のコンピュータのゼミの卒論なのである。だから，この小説にはゼミで勉強したと思われる内容が詰め込んである。それだけなら小説風の解体書になってしまうのだが，そうはなっていないところが，卒論として認めてもらえた所以だろう。

グリフォンズ・ガーデンというのは，主人公が大学院修了後に就職した研究所の名前である。そこには秘密裡に実用化されたバイオコンピュータがあった。主人公はそのバイオコンピュータに向かって，仮想の世界を創り始める。これがひとつのストーリー。主人公の現実世界と，主人公が自分の人生をアレンジして創り出したコンピュータの中の仮想世界が交互に描かれる。さらに，全体が現実世界と仮想世界それぞれの主人公の彼女，あるいは先輩（こちらも女）の対話を中心に描かれている。その対話も文学にありがちな，見るということは，聴くということは，自我があるということは，などというテーマなのだが，徹底的に理科系のアプローチなのがポイントだ。認知科学だろうか，そのあたりの知識をもとに，徹底的に理系な会話をさせ，仮説が渦巻き，女の論理で話がとぎれて，次のターンへいく。たとえば，文章を音で認識するか/文字で認識するか，東京は多次元で札幌は2次元などなど，他愛のない仮説が飛び交う。ひょんなことに興味を持ちながらそれを客観的に語れ，知識も豊富な主人公の恋人は理系な思考人間の理想の女だ。卒論だけあって，小説としての完成度はいまひとつだし，専門用語がなんの説明もなく使われていたりするが，それなりに面白い小説だ。一応，派手な物語的展開を期待してはいけない，とだけはいっておこう。 (K)



バーチャル・
テック・ラボ
舘暲・廣瀬通孝監修
工業調査会刊
☎03(3817)4701
四六判 310ページ
2,880円(税込)

人工現実感とは，仮想環境をあたかも現実の環境のような感覚で体験したり，逆にその仮想世界でも行動することを可能とする新しい技術だ。一方，テレイグジスタンスとは，従来の時空の制約から解放され，それらにとらわれない仮想環境に存在することを目指す新しい概念だ。本書は，これら2つの分野で世界的に活躍している研究者9名が，同じ研究所で働いていると仮想的に設定し，それぞれの専門分野からの研究成果の報告書というかたちで構成されている。多少ブッ飛んでると思える箇所もあるが，現在の科学がどのくらい進歩しているかを見るのには適しているだろう。

すみやす荘
パソコン通信日記
泉名文子著
啓学出版刊
☎03(3233)3731
B6判 186ページ
1,200円(税込)

文字で全部読ませるよりは，まんがで解説したほうがわかりやすいだろう，というわけなのか，とにかく「まんがパソコン入門」と題したシリーズのパソコン通信編である。本書では，パソコン通信を始めるにはまず何から揃えたらよいか，ということから入って，パソコン通信のノウハウを初心者にも絶対にわかるように書いた(描いた？)ものだ。まんがの内容は他愛のないものだが，「土台くん」「明日香さん」と，あの「めぞん一刻」をパロっているのが笑える。とはいえ，結局まんがだけでは説明しきれないので，ちゃんと説明書きの箇所もあるのだが。

Q **ゲームなどでときどき使われているテキストグラフィックデータを読み取るサブルーチンを作ったまではよかったのですが，実際に使ってみた結果なぜかありえないことが続出しています。TXSAVE.Cはテキスト画面を非圧縮でセーブするプログラムです。TXLOAD.Cは非圧縮データをグラフィック画面に展開するプログラムです。これらのプログラムはテストで作ったのですが，これが原因だったら最悪です。最後に問題のTPOINT.Sです。基本的にはグラフィック関数のPOINT(　)と使い方は変わらないはずなのですが，ここまでくると手のつけようがありません。使用しているハードはEXPERTで，システムは標準のHuman68k ver.2.0です。　埼玉県　竹内　正臣**

A 結論からいうと，TPOINT.Sに不備はありません。原因はTPOINT.Sにあるのではなく，テスト用に作成したTXSAVE.C，TXLOAD.Cにあるのです。XCやX-BASICでファイルをオープンする命令はともにfopenですが，XCではファイル入出力における改行コードの扱いをそのままにするか，変換するか2つのモードが選べます。

モードにはテキストモードとバイナリモードがあります。テキストモードでは入力時に¥r¥nを¥nに，¥nを¥r¥nに変換します。バイナリモードでは入出力時にデータ変換はありません。グラフィックデータの入出力時にデータの変換が行われては困りますから，ここではバイナリモードにしておかなければなりません。ですからTXSAVE.C，TXLOAD.C中にある，

fl=fopen("TEXT.DAT","r")
fl=fopen("TEXT.DAT","w")

は，

fl=fopen("TEXT.DAT","rb")
fl=fopen("TEXT.DAT","wb")

にしなくてはいけなかったのです。オープンモードについてはCライブラリマニュアルのfopen，fmodeを参照してください。

さて質問の回答はこれでおしまいですが，竹内さんはわざわざディスクにプログラムを収めて送ってきてくれたので，TPOINT.Sの一部を抜粋して掲載し，プログラムの参考になる話を付け加えたいと思います。

リスト3はX座標の値（d0レジスタ）とY座標の値（d1レジスタ）からテキストRAMのアドレスを計算し，その後d0レジスタにX座標÷8の余りを求めている部分です。テキスト画面は1ビットが1ドットに対応していますので，1バイトで8ビットを表すことができます。また1ラインは1024ドットなので，1ラインに含まれるバイト数は1024÷8=128バイトとなります。したがってテキスト画面のアドレス算出方法をBASIC風に書くと，

ADRS=$E00000+Y＊128+INT(X/8)

となります。式中$E00000はテキスト画面の先頭アドレスです。これをマシン語にすると次のようになるでしょう。

```
lea.l  $e00000,a0
lsl.l  #7,d1       *Y×128
move.w d0,d2       *X座標をd2に保存
lsr.w  #3,d0       * INT(X/8)
adda.w d0,a0
adda.l d1,a0       *a0=アドレス
```

d0,d1レジスタが破壊されますから，あとでX÷8の余りを求めるためにd2レジスタにX座標を保存しています。竹内さんは余りを求める方法がわからなくて，divu命令を使ったようですが，8の余りを求めるのに割り算は必要ありません。8の余りの範囲を考えてみてください。0～7ですね。つまり余りはd0レジスタの下位3ビットを取り出せばいいのです。X÷8の余りを求めるには単純に，

```
move.w  d2,d0  *d0をd2に復帰する
andi.w  #7,d0
```

でいいのです。

Q **SIONⅡで改めてMAGICの威力を思い知らされ，3Dばりのゲームを作ってやろうと意気込んだのはいいのですが，中盤でつまずいてしまいました。物体数の数が多くなり，白い線が飛び交って画面中がごちゃごちゃになったので物体の色の塗り分けをしようと思ったところ，これができないのです。SIONⅡではできているようなのでソースリストを見てみましたが，よくわかりません。MAGIC4の拡張機能も詳しく述べられてなかったので，できればそれと一緒にやり方を教えてください。　千葉県　畠山　尚**

A 付録ディスクに収録されSIONⅡに使われたMAGIC4（MAGIC ver.2.08）は，本誌91年9月号に掲載されたMAGIC ver.2.00にあったバグの排除，および処理速度を上げたもので，MAGIC4になって新たに拡張された機能はありません。物体の色の塗り分けの機能はver.2.00になって拡張された機能ですが，掲載号も含めて拡張機能の使い方を詳しく述べていませんでした。この場を借りて拡張機能の使い方を説明しましょう。その前にver.2.00から変更された箇所を紹介します。

●disp_flame.sの高速化

最近になってflameはframeの間違いだ

リスト1　TXSAVE.C

```
main()
{
int     x=0;
int     y=0;
int     s=0;
int     fl=0;

s=SUPER(0);

fl=fopen("TEXT.DAT","w");

for(y=0;y<=399;y++)
for(x=0;x<=511;x++)
fputc(t_point(x,y),fl);

fclose(fl);

SUPER(s);
}
```

リスト2　TXLOAD.C

```
main()
{
int     x=0;
int     y=0;
int     fl=0;

if((fl=fopen("TEXT.DAT","r")) == 'NULL') exit();
screen(2,0,1,1);
for(y=0;y<=399;y++)
for(x=0;x<=511;x++)
pset(x,y,fgetc(fl));
fclose(fl);
}
```

リスト3

```
moveq.l #10,d2
lsl.l   d2,d1                  *        Y*1024
add.l   d0,d1                  *        Y+=X
divu.w  #8,d1                  *        Y/=8
move.l  d1,d0
lea     $e00000,a0
and.l   #$ffff,d0
add.l   d0,a0
move.l  d1,d0
swap    d0
and.l   #7,d0
```

リスト4

```
  1: *
  2: * MAGIC拡張機能サンプル
  3: *
  4:
  5:          .include         iocscall.mac
  6:          .include         doscall.mac
  7:
  8: __AUTO:           equ     $FD13
  9:
 10:          .text
 11:
 12:          move.w  #-1,d1
 13:          IOCS    _CRTMOD
 14:          move.w  d0,crt_mode
 15:
 16:          lea.l   init_data,a0
 17:          dc.w    __AUTO
 18: loop:
 19:          clr.w   d1
 20:          IOCS    _BITSNS
 21:          btst.l  #1,d0
 22:          bne     prog_end * ESCが押された
 23:
 24:          lea.l   square_data,a0
 25:          move.w  #%11_000_000,color  * 緑
 26:          dc.w    __AUTO
 27:          lea.l   buttai1,a0        * 物体1
 28:          addq.w  #1,pitch1
 29:          dc.w    __AUTO
 30:
 31:          lea.l   square_data,a0
 32:          move.w  #%00_111_000,color  * 赤
 33:          dc.w    __AUTO
 34:          lea.l   buttai2,a0        * 物体2
 35:          subq.w  #1,pitch2
 36:          dc.w    __AUTO
 37:
 38:          lea.l   display,a0        * 表示
 39:          dc.w    __AUTO
 40:
 41:          bra     loop
 42: prog_end:
 43:          move.w  crt_mode,d1
 44:          IOCS    _CRTMOD
 45:          move.w  #$ff,-(sp)
 46:          move.w  #6,-(sp)
 47:          DOS     _KFLUSH
 48:          addq.l  #4,sp
 49:
 50:          DOS     _EXIT                  * _EXIT
 51:
 52:          .data
 53:
 54: crt_mode:
 55:          ds.w    1
 56: init_data:
 57:          dc.w    18     * init
 58:
 59:          dc.w    17     * scrmod
 60:          dc.w    1+256 *512*512(拡張モード)
 61:
 62:          dc.w    16
 63:          dc.w    255
 64:
 65:          dc.w    7        * set mode
 66:          dc.w    2        * pset
 67:
 68:          dc.w    21       * depth
 69:          dc.w    20
 70:          dc.w    2000
 71:
 72:          dc.w    11
 73:          dc.w    0
 74:          dc.w    0
 75:          dc.w    11
 76:          dc.w    1
 77:          dc.w    0
 78:          dc.w    11
 79:          dc.w    2
 80:          dc.w    80
 81:          dc.w    11
 82:          dc.w    3
 83:          dc.w    0
 84:          dc.w    11
 85:          dc.w    4
 86:          dc.w    0
 87:          dc.w    11
 88:          dc.w    5
 89:          dc.w    0
 90:          dc.w    11
 91:          dc.w    6
 92:          dc.w    0
 93:          dc.w    11
 94:          dc.w    7
 95:          dc.w    0
 96:          dc.w    11
 97:          dc.w    8
 98:          dc.w    0
 99:
100:          dc.w    15
101: square_data:
102:          dc.w    12          *コマンド 3D_DATA
103:          dc.w    4           *頂点の数
104:          dc.w    -10,15,2    *頂点リスト
105:          dc.w    -10,-15,2
106:          dc.w    10,-15,2
107:          dc.w    10,15,2
108:          dc.w    4           *線分の数
109: color:   dc.w    255         *物体のカラーコード
110:          dc.w    0,1         *線分リスト
111:          dc.w    1,2
112:          dc.w    2,3
113:          dc.w    3,0
114:
115:          dc.w    15
116:
117: buttai1:
118:          dc.w    11
119:          dc.w    0
120:          dc.w    20          * X
121:          dc.w    11
122:          dc.w    7
123: pitch1:  dc.w    0           * ピッチ
124:          dc.w    13          * 変換
125:          dc.w    15
126:
127: buttai2:
128:          dc.w    11
129:          dc.w    0
130:          dc.w    -20         * X
131:          dc.w    11
132:          dc.w    7
133: pitch2:  dc.w    0           * ピッチ
134:          dc.w    13          * 変換
135:          dc.w    15
136:
137: display:
138:          dc.w    14          * 表示
139:          dc.w    15
140:
141:          .end
```

と気づきました。スペルミスってすごく恥ずかしい。今後，プログラムを発表する機会があればスペルをframeに直す予定です。裏画面の消去方法を変更しました。

●line.sのバグ潰しと高速化

ラインの傾きが1に近いときに，線がつながらないバグを取りました。高速化は力技の256倍ループ展開！（ver.2.00は8倍ループ展開だった)。これにより256×256ドットモードでは一度もループを回さずに，ラインを表示することができます。

●その他のバグ

拡張モードでプレーンを4枚使用する場合に，優先順位を変えると表示がおかしくなりました。優先順位を変更した場合でも，ちゃんと表示できるように訂正しました。

3D−2D変換で求めた表示X，Y座標の下位1ビットを切り捨ててました。ああ顰蹙を買ってしまいそう。妙に動きが粗いと思った方もいたでしょう。すみませんでした。MAGIC4では修正してあります。

では拡張機能の使い方を説明します。

拡張機能を利用したプログラムがリスト4です。実行すると画面には赤と青の四角形がクルクルと回ります。プログラムではひとつの物体形状データを，拡張機能を使って色分けして表示しています。このリストを見ながら解説を進めていきましょう。

まず，拡張機能を使うことをMAGICに指定するには，画面モードの設定（コマンド番号17 CRT）でパラメータに$100（256）を足すことで指定します。リストでは60，61行が画面モードの設定になっています。

物体ごとの表示色の指定は3D物体定義（コマンド番号12　SET 3D DATA）で行います。リストでは101～115行が物体の形状データです。SET 3D DATAコマンドに続けて頂点数，頂点リスト，線分数とここまでは通常のパラメータが並びますが，線分リストを定義する前に，物体の表示カラーコードを指定します（ラベルCOLOR)。リストに注釈をつけておいたので，そちらを見てもらえばパラメータの指定の方法はわかると思います。　　　（影山　裕昭）

図1

$000C SET 3D DATA
物体の3Dデータを設定する

〈データ〉	〈バイト数〉	〈内容〉
$000C	2	コマンド
PCT	2	頂点の数（n）
X_1	2	
Y_1	2	頂点１
Z_1	2	
		頂点リスト
X_n		
Y_n		頂点n
Z_n		
LCT	2	線分の数（m）
COLOR	2	拡張モード時のみ指定
LS_1	2	線分１
LE_1	2	
		線分リスト
LS_m	2	線分m
LE_m	2	

＊ひとつの線分は2つの頂点ナンバーで指定する

質問にお答えします

日ごろ疑問に思っていること，どんなことでも結構です。どんどんお便りください。難問，奇問，編集室が総力を挙げてお答えいたします。ただし，お寄せいただいているものの中には，マニュアルを読めばすぐに解答が得られるようなものも多々あります。最低限，マニュアルは熟読しておきましょう。質問はなるべく具体的に機種名，システム構成，必要なら図も入れてこと細かに書いてください。また，返信用切手同封の質問をよく受けますが，原則として，質問には本誌上でお答えすることになっていますのでご了承ください。なお，質問の内容について，直接問い合わせることもありますので，電話番号も明記してくださいね。

宛先：〒108 東京都港区高輪2-19-13
NS高輪ビル
ソフトバンク株式会社出版部
「Oh! X質問箱」係

STUDIO X ON AIR

FROM READERS TO THE EDITOR

待ちに待った夏がやってきました。静かに冷房の効いた部屋で涼むのもいいですが，スカッと晴れた日には，外に出て暑さなんてふっ飛ばすぐらいの気力で，ガンガン遊んでしまいましょう。でも日射病には気をつけようね。

◆ひさびさの付録ディスクにふるえています。なんと解凍後5枚になるなんて，うるうる……。おかげでフロッピーディスクを買いに走って，1,760円の6月号になってしまいました。ボーナス前は非常に苦しい。　神山　卓裕(34)静岡県

苦しい思いをしたぶん，付録ディスクを楽しんでいただけましたか？

◆すごい，PCM8はすっごいです。感動しました。FM音源8音にPCM8音というのは，すごいのひと言としかいいようがありません。大変なものをありがとうございました。

熊下　泰章(17)岩手県

本当にPCM8を制作した江藤氏には，頭が下がる思いです。熊下さんも感謝の気持ちを込めて，PCM8を活用しましょう。

◆とうとうPCM8が付録に付いたということは，LIVE in '92でも要PCM8というのが増えていくのだろうか。また，内蔵音源＋PCM8＋MIDIというものも出てくるのだろうか。

村上　輝彦(19)香川県

さっそく今月号で登場していたりします。

◆「SIONⅡ」感動しました。3Dシューティングはあまり好きじゃなかったのですが，このゲームは演出も凝っているし，音楽も素晴らしいし，スピード感もあって楽しめました。エンディングで“SEE YOU AGAIN IN THE NEXT GAME?”の“?”が気になりましたが，次回作も期待していますのでがんばってください。

伊藤　久恭(17)埼玉県

がんばります(M.H)。

◆PCM8のようにいままで“ハード的に不可能”と思われていたことを実現するソフトってすごい。そのうち1600万色同時発色を可能にするソフトも出るに違いない（無理だって）。

坂井　純一(22)茨城県

7月号で紹介したTOYBOXでは，209万色表示を実現。不可能ではなかったりして。

◆10周年おめでとうございます。私のマイコン歴も（56歳の定年後から！）ポケコンPC-1500から始まり，X1（マニアタイプ），X68000（初代）と10年以上続いています。1983年頃の本誌表紙にあったMZ,X&ポケコンシリーズが懐かしい。これからもシャープさん，ソフトバンクさんの努力に負けないよう，死ぬまでついていきたいと思いますのでよろしく。中野　譲(66)兵庫県

こちらこそよろしくお願いします。これからも末長くお付き合いしていきましょう。

◆僕は1986年5月号からの読者ですが，ふと思い返してみるとそのころのOh!MZの精神というか，雰囲気というかそういうものが現在まで受け継がれていて“あーOh!Xっていいなあ”と思います。ちなみにそんなころからの読者ですので，山田さんとか高橋さんとかの学生時代に出したハガキも知ってます。へへ。

田間　豊常(17)兵庫県

さて，なんのことでしょう。僕は過去を振り返らない人間だから，過ぎたことは忘れてしまいましたよ（知らんぷり）。

◆まずは創刊10周年おめでとうございます。私がOh!Xを買い始めたのは，Oh!MZからOh!Xになった1987年12月号ですから早いものでもう4年と6カ月が過ぎました。これからも“1家に1冊，あると安心Oh!X”をモットーにがんばってください。それから，今年は私も学生からサラリーマンにクラスチェンジしました。今年を境に私もさらなる飛躍をとげたいと思います。

長石　裕行(23)広島県

そうなると発行部数で少年ジャンプを超えろ！　となってしまいますね。

◆8年前からのお付き合い，長いようで短いものです。ドイツのあのお方はお元気なのでしょうか。また，あの独特のゲーム評を読みたいものです。ところで，先日単語帳を開いたとき“attach”を見かけました。当時「こんなモン誰もわかるわけない」と思っていたその単語も受験生にとっては，必須だったのですね。

今井　礼治(21)神奈川県

それでも，きちんと最後までたどり着く人がいたんですからすごいものです。

◆Oh!MZ/X10年間の歩みは面白かった。私が初めて所有したパソコンはX68000でしたので，その昔ビジネスパソコンなどといわれていた8ビットマシンを見ていると，すごく昔の話だなと錯覚してしまいます。実はたったの10年なのに。人間って思ったよりすごいものだ。

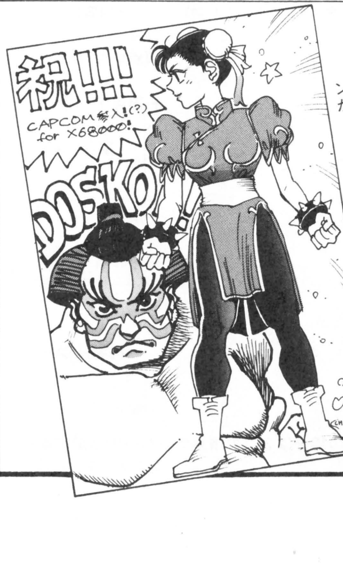
五十嵐　豊(24)千葉県

日進月歩のコンピュータ業界ですから，10年後はどうなっているんでしょう。

◆私がまだMZ-2000の「ちゃっくんぽっぷ」に興じていたころ，3重和音の記事が目に付き打ち込んだことがありました。やけに音痴だったり，ビブラートというよりはディチューンのようだったり，無謀にも“マジカルサウンドシャワー”を演奏させようとして自爆したり，とさんざんでした。しかし，曲を演奏させること自体がすごく面白く，パソコンをゲーム以外の目的で使用するきっかけとしては十分でした。Oh!X（MZ）は，自分にとってまさに“パーソナルコンピューティングへのひとつの手助け”となったわけですね。　中島　民哉(21)埼玉県

う～ん，なんて心強いお言葉。

◆10周年おめでとうございます。思えば以前，京都の会社に勤めていたときに会社の人がOh!MZという雑誌を持ってきていて，私はなんの雑誌か知りもしないで見せてもらったことがありました。そのころは「上海」というゲームの名前さえ知らなかったのに……いまではすっかり



◀橋本　和典　東京都
心はすでに次回作，の気持ちがありありと見えますね。「ファイナルファイト」に全力投球のカプコンが，どう応えてくれるか楽しみです。

◀板垣　央　千葉県
ゲームタイトルに自分の趣味をふりかけたという感じのイラスト。医者の傍らにこんな看護婦が仁王立ちしていたら……怖い考えになってしまった。

◀丸藤　俊之　神奈川県
う～ん，さすがイラスト常連の丸藤さん。丸みを帯びたSIONⅡのデザインがかっこいいです。これからもがんばって投稿してください。

◀松村　知己　石川県
プレイしながら体が避けている女の子がポイント。ゲームの雰囲気がよく伝わってくるのがいいですね。

読者になっているとは。すみずみまで理解しているなんてことはありませんが，忘れたころに役に立つ記事が多くて溜まっていくぞ。

谷口　生美(28)京都府

これからもいろいろな障害があるでしょうけれど，読者ともどもがんばっていきますので，よろしくお願いします。

◆編集部の方々はアンケートハガキを1カ月に何枚くらい読んでいるのでしょうか。ご苦労さまです。なんでしたらストレスを発散させるために，このハガキを破り捨ててください。気持ちがいいかもしれませんよ。さあ，どうぞ。私の気が変わらぬうちに。堅田　啓之(21)千葉県

ちなみに返送されてきた6月号のアンケートハガキは，重さ約4870g，積み上げたら約43cmありました。もちろん全部目を通していますからね。

◆前田編集長が特別寄稿の原稿中で「読者ハガキが1983年3月号あたりから……」と述べられていますが，私もMZ-700を手に入れその号から購読を始めたのでした。すると，もう9年以上も継続して購読していたのか！

増田　秀樹(26)東京都

9年なんて考えると，あらためて月日の長さを感じてしまいます。

◆私がOh!Xに出合ってからまだ数カ月ですが，「10年間の歩み」を読んでいると「ああ，同じ時代を生きていたんだなあ」と親近感を感じてしまいます。そういえば，昔X1を見せてもらい，当時靴の裏ユーザーであった私は激怒したのを思い出しました。「PC-8801はなあ，ペイントしている間にコーヒーが飲めたんだぞ(笑)」。

井口　和幸(23)静岡県

そういえば電源部でホットミルクが作れるとか，目玉焼きが作れるすごいマシンもあったなあ。

◆Oh!MZ創刊数カ月後から毎月欠かさず購読している私は，「OLD TIMES Oh!MZ,Oh!X」をたいへん懐かしく読ませていただきました。その中にあった「Oh!Xの誕生」で「5年前に出された問題って何？」というボケをかました人は，何をかくそう私の女房です。Oh!MZ/Xの歴史の数行を飾れたことを夫婦ともども喜んでいます。

中野　春一(31)東京都

本当に喜ばれているのかちょっと不安です。

◆う～む，安井百合江さんのデビューは私が初めてOh!Xを買ったときだったのか。ところで，私の妹も「百合江」というんですね。まあ，安井さんのレベルとまではいかなくても，せめてパソコンに興味を持ってくれたらなあと思い，私は機会があるたびにX68000の機能をアピールしたり，Oh!Xの愉快な記事を見せたりしました。しかし，なんの効果もなく，結局妹はまっとうな道を歩んでいくようです。これは兄として喜ぶべきことなのか，それとも悲しむべきことなのか。ちなみに妹には荻窪圭という女の子の友達がいます（本当）。　松本　拓司(18)埼玉県

これは本当に怪しい。本当と強調するところがさらに怪しい。

◆「第3回Oh!Xアンケート分析大会」のベストライターに名前が出ていなかったので，声を大にしていいたい。私には難しいながらも，有田隆也さんの連載は毎回面白く読ませてもらっています。それから，祝一平さんの連載は復活しないのですか。「人類タコ科図鑑」なんかは危険なんでしょうか。　滝本　哲之(18)埼玉県

現在，社長として多忙の身ですが，そのうち何かしでかしてくれるかもしれません。ちょっとだけ期待していましょう。

◆そうか，毛内氏は今月号でライターをやめちゃうのか，う～んさみしいなあ。(で)氏はいきなり放浪の旅に出るっていうし。まだ若いのにいったい何があったんだろう。まさか，ギミア・ぶれいくでやっているアニメ「さすらいくん」を見て旅に出たくなったとか(笑)。早く帰ってきてくれ～，(で)氏のいないOh!Xなんて。

青島　一高(23)静岡県

完全に引退したわけではないので，静かに復活を祈っていてください。

◆いやあ，だまされていた。まっさか荻窪さんと吉田さんが同一人物だったとはねえ。全然吉田さんのカラーが出てないんだもん。でもいちばん笑えたのはやっぱり祝さんだね。いい味出しすぎ，超面白すぎる。あーよく笑ったわ。

内藤　陽一(25)東京都

おなかがよじれるほど笑ってもらって，祝社長も喜んでいることでしょう。

◆私は吉田幸一氏が書く文章のファンでした。ですから，吉田氏の真似をしている（と私は信じていた）荻窪圭氏には，あまりいい感情を持っていませんでした。数カ月前，久しぶりに吉田氏の名前を見つけたときも「やはり本物は違う」などと喜んでいたんです。ああ，それなのに……。　木下　孝雄(20)東京都

真実ほど残酷なものはない，諸行無常は世の常といったところですか。

◆いままで飛ばし読みしていた「ハードウェア工作入門」をじっくり読んでみました。あらためて読み直してみると，とてもていねいでわかりやすいですね。これならよい復習になります。これから基本へ戻るそうですが，期待していますよ。　永井　正彦(24)福井県

読むだけでなく実践してみてくださいね。

◆僕のかわいいX68000は，たまにアクセスしたディスクの内容をふっ飛ばしてくれます。これって病気なのかなあ。　西馬　由岳(17)兵庫県

確かに病気です。潔くシャープ病院へ入院させたほうがいいでしょう。

◆祖母がハワイへ行ってきた。おみやげはもちろん“マカダミアナッツ”と思いきや，“DA-KINES”というチョコレート菓子($5.25)。どんなものかというと，さきイカ10本ぐらいを束にして一方の端をミルクチョコレートで固めたものなんです。さすがに口にする勇気が出るまで時間がかかりましたが，意を決してひと口頬張ると……なんだ食べれる味じゃないかと思えました。しかし1分後，うぇぇあ～，あと味が……。もうハワイアンは信じられない（笑)。

小川　幸泰(19)北海道

以前，アメリカ帰りのT.F氏が電気コードにそっくりなお菓子を買ってきてくれました。これもなかなかきてる味で，ちょっとばかりアメリカ人の味覚を疑ってしまいました。

◆最近「STUDIO X」にも“怪しい～”という言葉が進出してきたみたいですね。私が初めてこの言葉をOh!Xで見たのは，某月某日に命からがら逃げ帰った人の文でしたが，ハード屋さんでは以前からこの言葉が使われていたそうです。ところで，そのうち「Z80ボード」とかいう怪しげな回路図をお送りするつもりですので，お覚悟を。　荒居　光徳(27)東京都

こちらとて並みの怪しさではたじろぎません。

◆店頭デモ記述言語を作りました。ストロボアクション，パン，フェードアウト，半透明など次々にCGを表示していくさまは圧巻。効果音や音楽との同期もでき，使い方も簡単です。マイコンテック四日市でデモをしているので見に来てください（毎週デモ内容は変えてあります）。

下田　達也(25)三重県

面白そうだけど三重県まではちょっと遠いですね。

◆「バトルテック」が出る。しかもPC-9801用より2カ月も前にだ。たぶんRPGルールよりも前だ。もしかするとボードゲームよりも前にだ。こ

れは買いだ！ ただし受験が終わってから……。
石井　大輔(17)東京都

ちょっとさみしいけど，やることはやらなきゃ。

◆LOGINで「大戦略IV」を見たとき「こ，こいつあ～すげえ」と思ったのはいうまでもありません。リアルタイムオペレーションに加え，通信による対戦プレイ，そしてあのリアルな戦闘シーン。これはぜひともX68000に移植してほしいと思いました。もちろんグラフィック，サウンドともにX68000では強化されたもので，スピードもそれなりにほしい。ああ，システムソフト（アルシス）さん，全力でX68000版を作ってくれないかな。
岡田　耕一(17)山口県

このビッグタイトルがどういう形で現れるか，期待して待ちましょう。

◆ごく最近，グリーンイグアナを飼い始めました。周りの人からはあきれられていますが，とってもかわいいです。そのイグちゃんの環境整備やビタミン剤のせいで，楽しみにしていた「ノア」はおあずけ。パソコンゲームにばかり夢中な私も困りものですが，イグちゃんに夢中になってく私はもっと怖い……。
福田　弘子(？)福岡県

がんばって惜しみなく愛情を注いで，ゴジラのようにでっかく育ててください。

◆島内のどこかにX68000ユーザーがいることは確かなのですが，いまだに会ったことがありません。Oh!Xを置いている本屋も４～５店は確認しました。６月号を初めて予約しようとしたときに，店の人に「１冊しか入らないのに先約があります」といわれてしまいました。チラッと名前を見たら「スーパー○○」とあったぞ。
石塚　孝之(35)新潟県

スーパーでOh!Xを購読している……わけでもないでしょうが，なんか不思議。

◆モデムを買った……電話した。彼女ができた……電話した。会社でミスった……電話した。ああ，みかか代があ！　藤井　実(21)千葉県

そして，電話代のために日々働くのですね。ううっ悲しすぎる。

◆X68000XVIを買いました。MSXそしてPC-8801とマスターしてきたので，なんとかなるだろうと思っていましたがディスクのフォーマットすらできませんでした。この５年間蓄積されたものが一気に崩れ落ちていくような感じで，とても悲しかった。　西山　直樹(18)福岡県

悲観的にならなくても絶対大丈夫。あせらずに，やりたいことを１つひとつこなしていきましょう。

◆社会人となって数カ月，とうとうイヤな上司と出会ってしまいました。彼の顔が「伝染るんです」の山崎先生にそっくりなのはいいんですが，人の話に首を突っ込んでは場をかき回すという癖があるため，皆から嫌われています。現在，この人物とは同じ寮に住んでいるため先行きが不安です（ああ会社員）。
鹿田　剛(20)東京都

世渡りが上手になるための試練と思えば，それくらいの障害なんて軽い軽い。

◆うちの大学にはOh!Xの読者が大勢います。だから昼休みが終わるころには，生協に大量に置いてあるOh!Xがなくなってしまいます。でも，僕は午前中の授業に出ていないから，余裕で手に入るのでした（おい，１年からこんな生活で大丈夫なのか？）。　荒田　圭哉(18)神奈川県

まあ，長老にならない程度にすべるのならハクが付いていいかもしれませんよ。

◆この間，MZ-700を8MHzにしました。CPUをZ80Eに変えて，パターンカットをしTTLをつないだところとりあえず動いています。8255がちょっと遅いせいか動きが怪しくなりますが，普通に使う分には十分。遅かったゲームはかなり速くなりました。MZ-700ってすごい。
金子　哲也(17)千葉県

そんなことをやってしまう金子さんもすごいですよ。

◆大学まで往復で５時間。部活にも入ってしまってから，毎日午前6時30分に家を出て午前１時に帰宅するため，眠るのは２～３時間。おかげでパソコンに触れる暇も何もない！ 誰か時間を１時間500円で売ってくれえ。
板垣　央(18)千葉県

500円で買えるのなら僕も時間がほしい。

◆先日，ラジオから「ゲストは星野美奈（当て字）さんです」と聞こえてきました。アイドルにうとい私は「そんな人がいたのかなあ」と思いながら納得してしまったのですが，何やら歌声が複数に聞こえるのです。最初は「変だな」と思っていたのですが，まったく気づかず，あとで本当は「ポチの皆さんです」といったことに気づいたのです。とんだ聞き間違いだったのですが，それにしても「たま」がいて「Mi-ke」がいて「ポチ」ですか……こうなると次は何がくるのでしょうかね。　藤原　彰人(22)岡山県

次はやっぱり「チョビ」，これに決まりでしょう。

◆うちのインコの「ピーちゃん」は「ピーちゃん，おはよう，こんにちは，こんばんは」としゃべります。Oh!Xを読んでいると端を噛んで穴を開けてくれます。　服部　弘治(18)京都府

今月号もピーちゃんの口に合いましたか？

◆先日，広島の深夜番組でマンガ特集がありました。そこで紹介された，広島の修○大学のマンガ愛好会の人たちが描いたイラストの中に，「田村だよ～」という文字が！ もしもあの田村君だったら，またOh!Xにイラストを描いてほしい。　宇都宮　勇夫(21)広島県

真実はいかに？ 田村さん元気にやっているのかなあ。

◆おまえたちには任せておけぬ，私自らが出る!! というわけで“ぐしぐし”は片手でやるものなんです。拳は目の下2cmに当てます。肱は広げません。顔は手を当てたほうの前方に軽く振るだけです。卑屈になったときよりも恥ずかしくなったときに使っております。ぐしぐし（←ほら）。このネタを使うとは愚かな……ふっふっふ。　岩瀬　貴代美(20)福岡県

これは不覚。今度から“ぐしぐし”を使うときには（©岩瀬貴代美）と付ける，ということで許してくださいね。

◆群馬のある中学校の中間テストの日程。

１日目　理科，社会
２日目　国語，数学
３日目　英語，田植え

須永　修司(16)群馬県

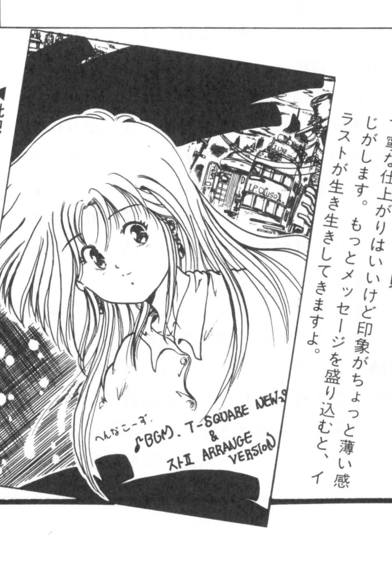
なんとも地域色あふれるテスト科目。田植えなんてどうやって採点するんだろう。

◆最近フジテレビの“皆殺しの数学”が面白い。あの数学の秋山先生が出ている番組で，数学嫌いの俺がこの番組のおかげで多少なりとも数学の面白さを知りました。数学って計算だけじゃないのですね。皆さんもぜひ見てみてください。
芝　幸一郎(17)東京都

要するに物事を見る視点を変えてくれたのですね。

◆ある日新聞を見ると“私女になりたいんですが……”という相談が載っていました。男よりは女のほうが楽だとか，いろいろな服を着れたり化粧できたりとか，そんなことを魅力に思ったのか20代の学生が相談していたわけです。それに対するある専門医の答えは“もし，女になったとしてもそれで満足しますか”ということでした。確かにそうかもしれませんが，どうでしょうかね。　大久保　敦(18)大阪府

◀枇杷木　亮　大阪府
やっぱり大声を張り上げて歌うとストレス解消にもなるし，すごく気持ちがいいですよ。実力が多少伴わないと近所迷惑野郎となりますけど。



◀横井　賢一　富山県
丁寧な仕上がりはいいけど印象がちょっと薄い感じがします。もっとメッセージを盛り込むと，イラストが生き生きしてきますよ。

やっぱり「隣の芝生は青い」ということなんでしょうか。男だろうと女だろうと嫌なことはあるんだし，変わったところでどうにでもなるものじゃないと僕は思います。

◆「性格や特技が正反対の２人が惹かれ合うのは，ごく自然なこと」と恋愛研究家(?)の金子俊一氏は語ったそうですが，実際は「似たもの夫婦」のような場合が多いそうです。先ごろ結婚した種ともこさんは，ダンナのことを「前世は双子だったに違いない」というくらい同じ性格なのだそうです。　野田　敏之(20)神奈川県

しかし，似すぎていると自分が２人いるみたいで疲れそう。

◆ガキのころから鉄道が好きだったのですが，最近よく思うことがあります。JRグループになってはや５年。ほとんどの車両には，JRマークがついていて東京駅などでもJR東日本，JR東海の新幹線が顔を並べています。しかし，結局は別会社だと思うのが，なんか悲しくなっちゃったりするのです。あと，東北新幹線の特急が消えてしまうのも悲しいものです。こんなことを考えるのは僕だけだろうか。

墨谷　雄二(21)栃木県

これもまた，時代の移り変わりなんでしょう。そういう気持ちは，思い出として大切にしまっておきたいですね。

◆10周年おめでとうございます。話は変わりますが，この間学校の授業中あまりにもうるさかったため，先生が超怒りました。そして，授業終了のチャイムと同時にいったセリフが「もうカンカンプーだ！」。いまだにこの捨てゼリフには，衝撃の余韻があります。

志田　健(16)東京都

ハガキを読んでいて思わず噴き出してしまいましたが，その先生もかわいそう。７代末まで語り継がれたりして。

◆５月16日の土曜日。私はいつものように宅配便のアルバイトをしていました。その昼休みに構内の荷物置き場をうろついていたら，X68000 XVIを見つけたのです。PC-9801，Macintoshはごろごろしているのですが，X68000は結構珍しい。なにげなく宛先を見たらOh!X編集部様と書いてあるではありませんか。すかさず班長のところへ飛んでいき，「この荷物を私に運ばせてください」と哀願し承諾を得ました。こうして月曜日配達予定の自分の受け持ちでもない荷物を持って，Oh!Xに合法的に侵入することに成功したのです。編集部内はいろいろな意味でミラクルワールドでした。詳細は内緒，うひひ。

柳沢　寿治(21)東京都

そんないい方をすると，Oh!X編集部は人跡未踏のジャングルみたいではないですか。それにしても見上げた根性，少しだけ感心しちゃいました。

▲横山　雅明　北海道
シムーミンやカールビンソンの世界にいそうな，シュールなレミングスたち。かわいいけれど難しい「レミングス」，皆さんは遊んでみましたか。

ぼくらの掲示板

●掲載ご希望の方は，官製ハガキに項目(売る・買う・氏名・年齢・連絡方法……)を明記してお申し込みください。
●ソフトの売買，交換については，いっさい掲載できません。
●取り引きについては当編集部では責任を負いかねます。
●応募者多数の場合，掲載できない場合もあります。
●紹介を希望されるサークルは必ず会誌の見本を送ってください。

仲間

★X68000ユーザーを対象とするサークル「LUM'S COMPANY」では，新規会員を募集します。毎月１回，DM形式の会報を発行しています。このサークルは全員参加で会報は門外不出となります。その代わり会費はありません。ゲーム作成にあたりプログラム，音楽，絵，企画のできる方を歓迎します。また，積極的に活動してくださる方に限ります。興味を持たれた方は，下記まで返信用切手を同封の上お申し込みください(RAMS-NETのLUM2へmailでもかまいません)。数に限りがありますので早めにお願いします。〒651-21　兵庫県神戸市西区伊川谷町有瀬254-2-406　伊東　智則(20)

売ります

★カラーインクジェットプリンタ「IO-735X-B」＋デモ版「バナナプリント」を送料込み115,000円で売ります。箱，付属品，マニュアルすべて揃っています。1991年10月に購入してから10回くらいしか使用していません(保証期間内)。連絡は往復ハガキに住所，氏名，年齢，電話番号を明記のうえお送りください。〒206　東京都多摩市関戸3-19-1-B303　木島　洋一

★X68000用1Mバイト増設RAMボード「CZ-6BE1A」を1,000円以上，アイテックのハードディスク「iT-X680(B)」を20,000円以上，Roland「MT-32」(旧型)を10,000円以上で売ります。すべて完動動品で，マニュアル，箱も揃っています。連絡は往復ハガキでお願いします。〒441　愛知県豊橋市牧野町149-5　岡本　昌泰(29)

★Roland「MT-32」を30,000円くらいで売ります。箱，付属品，マニュアルありで完動美品です。官製ハガキに希望価格と電話番号を明記のうえ連絡してください。また，X68000 Compact XVI用2Mバイト増設RAMボード「CZ-6BE2D」と交換も可。その場合は当方が10,000円プラスします。〒192　東京都八王子市中野山王3-20-247-2　吉川　聡(20)

★シリアルプリンタ「MZ-1P14」(ケーブルはMZ,X1用のどちらかを付けます）を送料込み10,000で売ります。完動品，箱，マニュアルありで用紙も付けます。連絡は往復ハガキに電話番号を明記のうえお願いします。〒610-01　京都府城陽市平川野原33-39　代崎　正浩(32)

買います

★MZ-2500用のプリンタを20,000円程度，パレットボードを5,000円程度，マウスを3,000円程度で買います。連絡は官製ハガキでお願いします。〒464　愛知県名古屋市千種区春岡1-11-3　昌和荘１号室　鈴木　伸卓(22)

★MZ-2500用RAMファイル「MZ-1R37」を20,000円程度，1Mバイト増設RAM「MZ-1R36」を20,000円程度，拡張I/Oポートを5,000円程度で買います。連絡は官製ハガキでお願いします。〒332　埼玉県川口市中青木4-9-8　須田　泰弘(22)

★X68000用拡張I/Oボックス「CZ-6EB1」を40,000円程度で買います。取扱説明書を付けてください。連絡は官製ハガキか封書でお願いします。〒572　大阪府寝屋川市成田東町13-4　比山　登(29)

バックナンバー

★Oh!X1988年12月号，1989年５月号を各2,000円(送料込み)で買います。折れや汚れはかまいませんが切り抜きは不可です。連絡は往復ハガキでお願いします。〒704　岡山県岡山市金岡東町3-1-4-4　柘植　嘉樹(20)

★電波新聞社出版の「オールアバウトナムコ(Vol.1)」を9,000円で買います。切り抜き不可でなるべく汚れのないものを希望します。連絡は往復ハガキでお願いします。〒408-03　山梨県北巨摩郡武川村上三吹345　輿石　学(21)

編集室から from E・D・I・T・O・R

DRIVE ON

このコーナーでは，本誌年間モニタの方々の意見を紹介しています。今月は6月号の内容に関するレポートです。第7期モニタの最後のレポートを紹介します。

●特別付録「創刊10周年記念PRO-68K」は，全体的にレベルの高い内容でそれ自体は好ましいと思いました。しかし，私はどうもいまひとつという印象が残っています。ディスクには，なんらかのデータもしくはある特定のデータを作成するツールが中心であったためです。もっともそれらのデータ形式（アニメにしろ音楽にしろ）に興味を持てばたまらないものでしょうが，やはりそれだけではちょっと……という気がします。ディスクの中では「SIONⅡ」のスピードに圧倒されました。まったくすごい，のひと言につきます。まだクリアしていませんが，3Dの可能性を強く印象づけるゲームでした。

宍戸　輝光(18)X68000 PRO,MSX2　東京都

●大人のためのX68000「第3回Oh!Xアンケート分析大会」によると，今年も祝一平氏が人気ライターの3位に入ってきているとか。最近は，Oh!X誌上で「I.I」の文字を見ることがなくなったというのになかなかの健闘ですね。やはり昔の読者からの組織票が効いているのでしょう。また，荻窪圭氏や西川善司氏の台頭は，両氏の活躍を如実に表しています。ほかのライターの方々も頑張ってください。来年のアンケートでは，祝氏なんか「その他」に含めてしまうくらいの活躍を期待します。そして，特別企画の中にあった「SENTINELとともに……」はS-OSの年表が載っていて，いままでの歴史を見せつけてくれます。現在，Z80のプログラムが掲載されることは驚くべきことです。他誌で8ビットCPUのプログラムなんてほとんど見られません。「THE SENTINEL」は「マシン語カクテルin Z80's Bar」とともに，Z80ユーザーに安心を与えてくれます。

中村　健(22)X68000　ACE-HD,MSX2, PC-386GS　埼玉県

●特別企画「Oh!MZ,Oh!X10年間の歩み」を読んで，なんだかいろいろなことがあったんですねえ，と感じました。10年前の1982年，私は確か小学校3年生でした。500円玉が発行され「E.T」が人気を博したのもこの年。私は今月で丸3年，4冊目の6月号と，この10年間のうち1/3しか読んでいません。でも，それなりに懐かしいものもあります。「対戦ポピュラス祝一平VS西川善司」とか「パンクローのバイナリ講座」とか「ぜんまいちゃん」とか。「パンクローのバイナリ講座」を読んで2進法を理解した私は，友人に2進法の数え方を教えてあげて，数学のテストのときに感謝されたこともありました。ところで，あの「安井百合江文明」って……恥ずかしいじゃないですか，本当にもう（プンプン，というわりに結構喜んでいたりするから，もう）。

安井　百合江(18)X68000 PRO　愛知県

●特別企画の「OLD TIMES」は本当に懐かしいことから，こんなこともあったんか！　と思うことまで楽しく読ませてもらいました。ただ，ところどころにある広告(？)の意味が不明です。特に霜降高原から，なんていっても知らない人には全然わからないでしょう。GEOSだってそう。ま，わからないのがまたいいのかもしれませんが。そして，特別寄稿は大爆笑!　Oh!Xでは久しぶりにお目にかかる祝社長の文章もイカすし，え～，荻窪圭と吉田幸一は同一人物なの？　とか，ありゃやっぱし皆さん結構MZな方なんだとか，とても楽しかったです。

松本　康弘(24)X68000 EXPERT-HD,X1turboZII,PC-286VS　広島県

●「対談！　GMコンポーザー」で，私もリクエストしたことのある古代さんとの対談が実現して嬉しいです。内容もたいへん面白かった。それにしても，やっぱり普段からいろいろな曲を聞いていらっしゃるのですね。まさかシカゴを聞いているとは……私も好きです（笑）。これからもこのコーナーを続けてほしいです。次はナムコの「めがてん」さんがいいですね。

功刀　和久(23)X68000 ACE-HD,X1turbo,PC-9801DA2　埼玉県

●付録ディスクにあったPCM8.Xはすごいです。すごいというよりよくやりましたね。僕なら思いついても，作る前からあきらめてしまうでしょう。本当にX68000の底力はすごい。また，特別企画は，当時を知らない私でも楽しく読ませていただきました。楽しむコツは，7%の知識と90%のノリ，そして3%の消費税ですね。

弦元　達也(21)X68000 ACE-HD

ごめんなさいのコーナー

5月号　COMMAND.OBJ

COMMAND.OBJ内で，スタックエリアが少なめに確保されているため，スタックエリアをアプリケーション側で確保していない一部のプログラムが，正常に動作しない可能性があります。

なお，S-OSではスタックエリアを自主管理する，という原則があります。つまり，システムスタックを使わず，アプリケーション自身にスタックエリアを用意して，プログラムを作成するようにしてください。

6月号 P.154 ペンギン情報コーナー

光カードシステムの問い合わせ先が“キヤノン”となっていましが，正しくは，“キヤノン販売(株)”です。関係者各位にご迷惑をおかけしましたことをお詫びいたします。

7月号　P.92　ヴェクザシオン

50行目のリストが間違っていました。正しくは以下のとおりです。

50　D$="t60～

P.171　ぼくらの掲示板

馬場英之さんの住所が間違っていました。正しくは以下のとおりです。本当に申し訳ありませんでした。

～吉祥寺南町1-28～

P.132　関数リファレンス

Small-Cリファレンスマニュアル中で正しくない表記が2カ所ありました。以下のように訂正してください。

●ビット演算子（NOT）

!a → ～a

●代入演算子

â=b → a^=b

バグに関するお問い合わせは
☎03(5488)1311(直通)
月～金曜日16：00～18：00

お問い合わせは原則として，本誌のバグ情報のみに限らせていただきます。入力法，操作法などはマニュアルをよくお読みください。

また，よくアドベンチャーゲームの解答を求めるお電話をいただきますが，本誌ではいっさいお答えできません。ご了承ください。

作りたいものが見つかったらプログラミングだ

▼特集では，プログラミングの魅力に取り憑かれた人々による，プログラミングの勧め，手法を紹介しました。記事を読むとわかりますが，大切なのは目的を持つこと。確かに，人によってプログラミングスタイルは違います。しかし，作り始める動機はただひとつ「作りたいものがあるから作る」のだと思います。これからプログラミングを始めようとしている人は，自分のやりたいことを見つけるように努力してみましょう。そうすれば自然と道は開けるはずです。

▼今月号では「Z'sSTAFF」とちょっと趣が違うグラフィックツール「MATIER」を紹介しました。エフェクト関係が充実しているこのツール。多機能のあまりひととおりの機能紹介だけでは，ちょと理解するのが困難かもしれません。来月号では，これらの機能を実際に使ったレポートをお送りしますので期待していてください。また，グラフィック関係では，「MIRAGE SYSTEM Model Stuff」の完成版によるレポートも行う予定です。

▼次に6月号で募集した「SIONⅡ」の各種音源版BGMテープの当選者を発表します（順不同，敬称略）。桑野拓也（東京都），山下禎久（静岡県），阪田弘喜（山口県），甲斐康彦（大分県），五十嵐紀博（北海道），折田貴弘（東京都），島津伸行（東京都），大内弘和（茨城県）ほか12名の皆様です。テープを聞いた感想をぜひお聞かせください。

▼そして，長いようで短い1年間のモニタ期間が終わりました。今月号で最後のレポートですが，これからも読者として元モニタの皆さんも積極的にアンケートハガキ，投稿で本誌に対するゲリラ活動を展開してください。9月号からは，心機一転新しいモニタの皆さんのご意見を紹介していきます。

▼今月号の「マシン語カクテルin Z80's Bar」は，著者の健闘空しく締め切りに間に合いませんでした。申し訳ありません。来月号では必ず復活しますので，楽しみにしていてください。

投稿応募要領

- 原稿には，住所・氏名・年齢・職業・連絡先電話番号・機種・使用言語・必要な周辺機器・マイコン歴を明記してください。
- プログラムを投稿される方は，詳しい内容の説明，利用法，できればフローチャート，変数表，メモリマップ（マシン語の場合）に，参考文献を明記し，プログラムをセーブしたテープ（ディスケット）を添えてお送りください。また，掲載にあたっては，編集上の都合により加筆修正させていただくことがありますのでご了承ください。
- ハードの製作などを投稿される方は，詳しい内容の説明のほかに回路図，部品表，できれば実体配線図も添えてください。編集室で検討のうえ，製作したハードが必要な場合はご連絡いたします。
- 投稿者のモラルとして，他誌との二重投稿，他機種用プログラムを単に移植したものは固くお断りいたします。

あて先

〒108 東京都港区高輪2-19-13 NS高輪ビル

ソフトバンク出版部

Oh!X「㋢㋮㊔」係

SHIFT・BREAK

▶わたしとA氏はとってもなかよし。泥縄で原稿を書いていると，横にきてはジャマをする。「Fさん，あのCM知っている？」やら，「Fさん，自転車につけるナビコンないかな」など。「Aさん，ヒマなの？」と私がいうと，「Fさん，頭ウスイね」。そんなことわかっているわい！　いつか締め切り前に原稿納めて，昇竜拳をお見舞いしてやる。 (FZ)

▶忙しい合間をぬって，浜松町の国際貿易センタービルの展望台に上がった。普段東京で暮らしているとこういう観光スポットには行かないもんだが，なかなかいい眺めだった。東京タワーとベイエリアを同時に見られるのはおトクな気分。しかし見渡すかぎりのビルの森だわ首都高は渋滞だらけだわで，AⅢかシムシティーなら失格の街だなこりゃ。 (浦)

▶私は深夜に白山通りを車で走ることが多いのだが，その通り沿いに深夜だというのに大勢の人間で賑わっているラーメン屋台がある。私はラーメンが大好き。その屋台のラーメンをいつか食べたいと思っているのだが，早く家に帰りたいあまりなかなか立ち寄る決心がつかない。誰か都内のうまいラーメン屋を教えて。 (H.K.)

▶8月7日あたりから北海道に行く予定。前から名馬オグリキャップを見てみたい，と思っていた。ついに実現するかもしれない。ほかに予定はないので，お勧めの場所を教えてください。北海道であればどこでもかまいません。愛車を持ち込むつもりなのですが，フェリーのチケットが取れないので，青森のほうまで東北道で行くことになりそう。 (S.K.)

▶今回の特集記事では教材用と称して中途半端なものを作ってしまったけど，SX-WINDOWアプリケーションプログラマの立場として，ああいうライブラリは本気でほしいと思っている。Toolbox似のシステムにToolkit似のライブラリ。ねえ，作ってもいい？　作ってもいい？　誰も作らないの？　誰も作らないのなら僕が作るよ……。 (A.T.)

▶SPA!にいわせると地雷女だそうで，平成の歩き方によるとカケヒキ女（天才型）だそうで，まあ，なんというか混乱したままの時代にだけは乗っていける人，自分の視点とメディアの視点が区別できなくなり，その歪みは抑え込まれているゆえに爆発するってわけで，容姿端麗な方に限り私が救済します（あれ？　何いってんだ，俺は）。 (K)

▼ゴジラ対モスラの小美人役が発表になった。国会議事堂でマユがかえるとか，悪のモスラが登場するとか，積極的に話題作りをしているゴジラ対モスラであるが，最大の焦点である小美人役が東宝の新人というのは拍子抜けだ。どうせなら，かつての小美人が生きていたという設定で，きんさんぎんさんを抜擢するくらいのことはやってほしかった。 (KO)

▶ちょっと前，SC-55を生活苦だった先輩から買いたたいた。ゲーム用の外部音源のつもりだったけど，少しぐらいは使ってやろうということで「SIONⅡ」ボスの曲をコンバートしてみた。「MMLってサイバーだなあ」の言葉を実感しながら作業を進め，ちょっと楽しさがわかってきたかなあ，というところで時間切れ。え〜んもっと遊びたいよう。(J)

▶6月7日，腕が少し黒くなった。多摩川サイクリングロードのせい。6月11日，腕にじんましんが出た。中華料理を食べにいったけど，特に思い当たるものはない。6月12日，またまた腕にじんましんが出た。前日との共通点を探ってみると，どうやら老酒のせい。ヘンなの，日本酒は平気なのに。6月28日，今日も腕が黒い。裏側と手は白いけど。 (A)

▶このところ作曲のお手伝いで友達の家に行くことが多い。バンドももちろん楽しいんだけど，こうやって姉妹同然の友達と作曲したり，音楽の話をしたりってのもなかなかよくて。「いつかこうやって2人でいたことも思い出になるのかな，でもその前に武道館でコンサートしようね」などと笑って話しているあたしらは，やっぱ脳天気なんだろな。 (E.O.)

▶荻窪圭＝吉田幸一の反響が大きい。Oh!Xの筆者は基本的に実名だ。ペンネームを使うのはそれなりに事情のある方だと思って間違いない（たいていは会社に内緒でがんばっている社会人スタッフ）。ペンネームを認めてくれという読者もときどきいるが，特に事情のない限り認める予定はない。匿名でなければできない話題など扱ってないはずだ。 (U)

▶やはり読んでおいたほうがいい。アラン・ケイの代表的論文がアスキーから出版されている。わかっているつもりの概念も，そのプロセスが頭の中で踊りだす。10年前，まだ初心者だったはずの恩師（Y氏）が「パソコンはメタメディアだ」と主張していたのを思い出す。氏は人が誰でもダビンチになれる世界を夢見ていた。世界はまだこれからだ。 (T)

microOdyssey

10周年という節目になにも書かせてもらえなかったので，ここで私にとっての「Oh!Xの10年」をまとめてみたい。

10年前，私もOh!MZの読者であった。6年前，就職のため上京，Oh!MZに配属される。

まもなく私は「Oh!MZはドラゴンだ」という言葉の意味を悟った。Oh!Xという本がドラゴンなのではない。編集長の安田氏こそがドラゴンなのだ，と（安田氏がソフトバンクを去り，誌名がOh!Xに変わってからは「ドラゴン」という単語自体がほとんど使われなくなった）。

とにかく，勢いはあるが無茶苦茶な雑誌だったOh!MZがかなりまともな雑誌になったのは氏の力によるものである。氏の影響が現在にいたるまで本誌のバックボーンとして存在する。宇宙戦艦ヤマトと沖田十三の関係のようなものだ。

現編集長T氏は難解な安田氏の思想を感覚的に理解できる凄い人であった。編集の@氏は安田氏の考えを実行できる凄い人であった。現在Oh!Dyna編集長のN氏はひとりで印刷屋，写植屋を切り盛りする凄い人であった。で，私は安田氏と@氏の下で修業中の新米編集者だった。ときには数行の文に十数回のリテークが出され，2ページの文章に1週間以上かかったこともある。これまでずいぶん無茶をやったが，いちばん胃を痛めたのはこの頃だ。

1987年初夏。Oh!MZは最大の危機を迎える。@氏の退社である。当時のOh!MZにおける特集がほかの記事とは別格であったように，特集担当の@氏の位置はまた特別だった。ある意味で@氏＝Oh!MZだったといっていい。

そして「質実剛健」という言葉もあまり使われなくなった。

いまだからいえるが，氏の抜けた穴を読者に悟らせないような本を作るというのは大変なことだった。残念ながら@氏は日本中から誰を連れてきても代わりのきくような人物ではなかったのだ。結局，特集補佐をやっていた私が特集要員となる。半面，それまでずっと担当だったS-OS関連記事のほうがどうも手薄になってしまったのは残念だった。

それでも仕事に慣れてきた1989年あたりの特集は我ながらノリがよいと思う。編集長T氏の下，体制が安定してきた時期だ。あとは新卒でも入ってくれば，もっといろんなことがやれる……と思った矢先に思いもよらぬN氏とよ嬢の異動。代わりに新規社員のS氏とE.O嬢が配属された。混沌のまま年末進行へ……。この時期はどうして本が出ていたのかよくわからない。

春。待望の新卒A君を迎え，入れ替わりにS氏が退社。1991年春。ライターあがりのJ君入社，……。

機種別情報誌でありながら，Oh!Xが目指しているものは特にマシンを選ぶわけではない。やれといわれればポケコンだろうが国民機だろうが同じ道を進むことだろう。ただ，個人的には，現在メイン機種となっている機械がX68000であったことは非常に幸運だったと思う。

Oh!Xの実体はひとつの思想として存在する。そういう意味では実にまっとうな雑誌である。方針がコロコロ変わったりはしない。10年を経て読者層は多少変わったが基本的にOh!Xは変わっていないこと，それが許されているということ。世の中そう捨てたもんじゃない。　（U）

1992年9月号8月18日（火）発売

特集　熱い数値演算の逆襲

Z's-EX用外部関数　ジャギー除去フィルタ

Oh!X LIVE in'92

恋をしようよYeah! Yeah!（LINDBERG）他

製品紹介

Communication SX-68K

SX-WINDOW開発ツールキット

バックナンバー常備店

東京	神保町	三省堂神田本店5F	03(3233)3312
	〃	書泉ブックマートB1	03(3294)0011
	〃	書泉グランデ5F	03(3295)0011
	秋葉原	T-ZONE 7Fブックゾーン	03(3257)2660
	八重洲	八重洲ブックセンター3F	03(3281)1811
	新宿	紀伊国屋書店本店	03(3354)0131
	高田馬場	未来堂書店	03(3200)9185
	渋谷	大盛堂書店	03(3463)0511
	池袋	リブロ池袋店	03(3981)0111
	〃	西武百貨店9F コンピュータ・フォーラム	03(3981)0111
神奈川	横浜	有隣堂横浜駅西口店	045(311)6265
	〃	有隣堂ルミネ店	045(453)0811
	藤沢	有隣堂藤沢店	0466(26)1411
神奈川	厚木	有隣堂厚木店	0462(23)4111
	平塚	文教堂四の宮店	0463(54)2880
千葉	柏	新星堂カルチェ5	0471(64)8551
	船橋	リブロ船橋店	0474(25)0111
	〃	芳林堂書店津田沼店	0474(78)3737
	千葉	多田屋千葉セントラルプラザ店	0472(24)1333
埼玉	川越	黒田書店	0492(25)3138
	川口	岩渕書店	0482(52)2190
茨城	水戸	川又書店駅前店	0292(31)0102
大阪	北区	旭屋書店本店	06(313)1191
	都島区	駸々堂京橋店	06(353)2413
京都	中京区	オーム社書店	075(221)0280
愛知	名古屋	三省堂名古屋店	052(562)0077
	〃	パソコンΣ上前津店	052(251)8334
	刈谷	三洋堂書店刈谷店	0566(24)1134
長野	飯田	平安堂飯田店	0265(24)4545
北海道	室蘭	室蘭工業大学生協	0143(44)6060

定期購読のお知らせ

Oh!Xの定期購読をご希望の方は綴じ込みの振替用紙の「申込書」欄にある　新規　継続』のいずれかに　をつけ，必要事項を明記のうえ，郵便局で購読料をお振り込みください。その際渡される半券は領収書になっていますので，大切に保管してください　なお，すでに定期購読をご利用の方には期限終了の少し前にご通知いたします。継続希望の方は，上記と同じ要領でお申し込みください。

海外送付ご希望の方へ

本誌の海外発送代理店，日本IPS（株）にお申し込みください。なお，購読料金は郵送方法，地域によって異なりますので，下記宛必ずお問い合わせください。

日本IPS株式会社

〒101　東京都千代田区飯田橋3-11-6

☎03(3238)0700

Oh!X　8月号

■1992年8月1日発行　定価600円（本体583円）

■発行人　孫　正義

■編集人　橋本五郎

■発売元　ソフトバンク株式会社

■出版事業部　〒108　東京都港区高輪2-19-13　NS高輪ビル

Oh!X編集部　☎03(5488)1309

出版営業部　☎03(5488)1360 FAX 03(5488)1364

広告営業部　☎03(5488)1365

■印　刷　凸版印刷株式会社

©1992 SOFTBANK CORP. 雑誌 02179-8 本誌からの無断転載を禁じます。

落丁・乱丁の場合はお取り替えいたします。

郵便はがき

料金受取人払

高輪局承認

1396

差出有効期間
平成6年7月
15日まで

108-00

507

(受取人)

東京都港区高輪
2-19-13 NS高輪ビル

ソフトバンク株式会社

Oh!X 編集部行

□□□-□□

電話

住所

氏名

年齢

職業・勤務先
学校・学部・学年

●編集部へのメッセージ

今月号の特集について	
いちばん良かった記事	興味のなかった記事
これから載せてほしい記事内容	本誌以外にお読みのパソコン雑誌
推薦する市販ソフト ソフト名： 推薦理由： 自分でプログラミングするとしたら，何を作りますか？	
あなたの愛機は(所有機種に○印をつけてください)　　ない X1(マニアタイプ,C,D,F,G,twin) X1 turbo(model 10,20,30,40,II,III,Z,ZII,ZIII) MZ-(80K/C, 1200, 700, 1500, 80B, 2000, 2200, 2500, 2861) X68000(初代,ACE,PRO,PROII,EXPERT,EXPERTII,SUPER,XVI,Compact, HD) その他　　MIDI楽器(　　) FD(　基)　TAPE　QD　HD(　MB)　MO　プリンタ(　　)	
年齢　歳　パソコン歴　年　男・女　プレゼントNo.	

切り取り線

通常払込料金
加入者負担

払込通知票

口座番号	東京	1	—	29307	金額	
加入者名	ソフトバンク株式会社				料金	払込み / 特殊 円
払込人住所氏名	※ （郵便番号 ）				備考	
					受付局日付印	

各票の※印欄は、払込人において記載してください。

記載事項を訂正した場合は、その箇所に訂正印を押してください。

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。（郵政省）

切り取らないで郵便局にお出しください。

通常払込料金
加入者負担

払込票

口座番号	東京	1	—	29307
加入者名	ソフトバンク株式会社			
金額				
払込人住所氏名	※			
備考		受付局日付印		

振替用紙

←点線から、きれいに切り取ってご使用ねがいます。

切り取り線

【定期購読のご案内】

●定期購読のお申し込みは、この郵便振替用紙のみとさせて頂き、銀行振込・現金書留によるご入金は、ご遠慮下さい。

●受付締切は、

1日発売 ： 発売日前月10日振込

8日発売 ： 〃 15日振込

15・18日発売： 〃 25日振込

です。

〈例〉 4月1日発売（Oh！PC 4月15日号）の場合、お振込の締切は3月10日です。

締切に間に合わなかった場合は、自動的に次号からの発送となります。

なお、すでに発売されているもの、また、お振込が締切に間に合わなかった月号のものは、定期購読ではお求めになれません。書店でご購入ください。

●定期購読誌のお届けは書店発売日より遅くなりますのでご了承下さい。

「発売日一覧」

◇毎月1・15日発売

Oh！PC

◇毎月18日発売

Oh！X

Oh！FM TOWNS

C MAGAZINE

Oh！Dyna

◇毎月8日発売

月刊情報処理試験

LANTIMES

THE WINDOWS

DOS／Vmagazine

切り取り線

切り取らないで郵便局にお出しください。

この欄は、加入者あての通信にお使いください。

送り先

お名前 フリガナ

お電話

ご住所 フリガナ 〒

定期購読申込書

□Oh！PC	年間（23回）	12,880円	（新規／継続 NO. ）
□Oh！PC	6ヶ月（12回）	6,720円	（新規／継続 NO. ）
□Oh！X	年間（12回）	7,200円	（新規／継続 NO. ）
□Oh！FM TOWNS	年間（12回）	7,440円	（新規／継続 NO. ）
□C MAGAZINE	年間（12回）	11,760円	（新規／継続 NO. ）
□Oh！Dyna	年間（12回）	9,120円	（新規／継続 NO. ）
□月刊情報処理試験	年間（12回）	8,160円	（新規／継続 NO. ）
□月刊情報処理試験	6ヶ月（6回）	4,080円	（新規／継続 NO. ）
□LANTIMES	年間（12回）	17,760円	（新規／継続 NO. ）
□THE WINDOWS	年間（12回）	11,760円	（新規／継続 NO. ）
□DOS／Vmagazine	年間（12回）	9,360円	（新規／継続 NO. ）

〔備考欄〕

この払込通知票は、機械で使用しますので、下部の欄を汚さないよう特に御注意ください。また、本票を折り曲げたりしないでください。 （郵政省）

満開の電子ちゃん

作・え 岡村祭

講読方法:定期購読もしくはソフトベンダーTAKERUでお買い求めいただけます。

★定期購読の場合=購読料6ヶ月分6,000円(送料サービス、消費税込)を、現金書留または郵便振替で下記の宛先へお送り下さい。

現金書留の場合:〒171 東京都豊島区要町1-19-3 いさみビル4F ㈱満開製作所

郵便振替の場合:東京 5-362847 ㈱満開製作所

● ご注文の際は、郵便番号・住所・氏名・電話番号を忘れずに記入して下さい。
● 3.5インチディスク版をご希望の方は、「3.5インチ版」とご指定下さい。
● 新規購読の方は「新規」と明記して下さい。なお、特に購読開始号のご指定がない場合は既刊の最新号からお送りいたします。
● 製品の性格上返品には応じられませんが、お申し出があれば定期購読を解約し残金をお返しします。

★TAKERUでお求めの場合=1部につき1,200円(消費税込)です。

● 定期購読版と内容が一部異なる場合があります。御了承下さい。
● お問い合わせ先 TEL(03)3554-9282(月~金 午前11時~午後6時)

(なお、定期購読版のバックナンバーについては定期購読の方のみご注文を承ります)

某月某日:タケルで間違って電脳倶楽部なるものを買ってしまう。

某月某日:起動する。いきなりやばい絵とやばい音楽が…。思わずリセットしてしまう。

某月某日:おそるおそる「今月のビープ音」というところをクリックする。家中に恥ずかしいサンプリング音が鳴り響く……。その日以来家族との会話がなくなる。

某月某日:ついに定期購読してしまう。だんだん、自分が自分でなくなっていくような気がした。

某月某日:電脳倶楽部が届く二、三日前からけいれんが起こるようになる。ああ、もうだめ……。

奥井一穂(石川県)

■店頭にて、新作ゲームソフト25〜30%OFF!!(税別)、超低金利オクトハッピークレジットをご利用下さい!!

■翌月末一括(8月末)払いOK!!手数料無料!!ご利用下さい。■店頭にて、新作ゲームソフト25〜30%OFF!!

■便利です。夜9時まで営業しております。お立寄り下さい。お待ちしております!!

パソコンプラザ OCT オクト

—オクトで始まるパソコンワールド—

☎ 03-3730-6271

●営業時間 AM 11:00〜9:00/日曜・祭日PM7:00 電話一本で、ハイ即納

〒144 東京都大田区蒲田4-6-7 FAX 03-3730-6273

全国通販

定休日:毎週火曜日
(祭日の場合翌日になります。)

店頭セール実施中

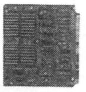
OCT-1 システム インフォメーション
- ▶全商品保証付(メーカー保証)
- ▶超低金利ハッピークレジット(1回〜60回)頭金ナシOK!
- ▶ボーナス一括払いOK!ボーナス 2回・4回・6回 払いOK!
- ▶配達日の指定OK!(万全なサポート体制)
- ▶商品の組合せ自由! オクトフリーダムシステム
- ▶店頭デモンストレーション実施中

特選周辺機器(送料¥500)

■SX-68MII MIDIインターフェイスボード
(システムサコム)¥19,800…特価¥13,500

■Fine Scanner X68(HAL研究所)
(HGS-68)¥39,800……特価¥24,500

■増設RAMボード=I・Oデータ



①PIO-6BE1-A(1MB)
¥25,000……特価¥15,800
②PIO-6BE2-2M(2MB)
¥50,000……特価¥31,000
③PIO-6BE4-4M(4MB)
¥88,000……特価¥54,000
④SH-6BE1-1M(1MB)
¥25,000……特価¥17,800

OCT-1 蒲田 来た〜!待望の夏のセール!! オトクに買って気分爽快!!

SHARP

X68000 PERSONAL WORKSTATION・XVI Compact

- ■16MHz■
- ■SX-WINDOW ver1.1■
- ■Attachment MEMORY BORD■

■CZ-674C-TN(定価¥298,000)

Ⓐ●CZ-674C-TN
●CZ-608D-TN(14型カラーディスプレイ)

定価合計¥392,800▶超特価¥283,000

12回	24回	36回	48回
¥25,900	¥13,700	¥9,500	¥7,500

Ⓑ●CZ-674C-TN
●CZ-607D-TN(14型カラーディスプレイTV)

定価合計¥397,800▶超特価¥285,000

12回	24回	36回	48回
¥26,100	¥13,800	¥9,600	¥7,500

Ⓒ●CZ-674C-TN
●CZ-614D-TN(15型カラーディスプレイTV)

定価合計¥433,000▶超特価¥310,000

12回	24回	36回	48回
¥28,300	¥15,000	¥10,400	¥8,200

Ⓓ●CZ-674C-TN
●CZ-606D-TN(14型カラーディスプレイ)

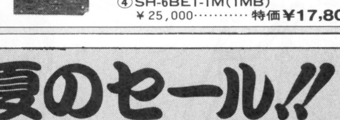
定価合計¥377,800▶超特価¥270,000

12回	24回	36回	48回
¥24,700	¥13,100	¥9,100	¥7,100

※送料¥2,000・税別
(クレジット価格は、送料・税込)

Compact限定セット

【送料¥3,000・税別 クレジットは送料・税込】



①CZ-674CH〈本体〉+CZ-608DH〈モニター〉+CZ-6FD5(5インチドライブ)

定価合計¥492,600▶超特価¥320,000

12回	24回	36回	48回
¥29,300	¥15,500	¥10,800	¥8,500

②CZ-674CH〈本体〉+CZ-606D〈モニター〉+CZ-6FD5(5インチドライブ)

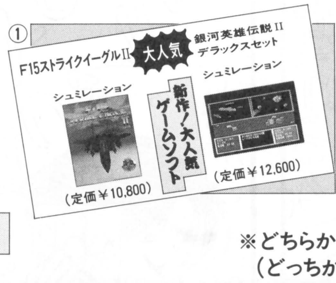
定価合計¥477,600▶超特価¥310,000

12回	24回	36回	48回
¥28,400	¥15,100	¥10,400	¥8,200

X68000 Compact 大好評記念プレゼント!!

—あなたのオクトから素敵な贈物—

今、Compactをお買い上げいただいた方は、プレゼントの①番か②番のどちらかをお選び下さい。プラス③番は、もれなくプレゼント!!



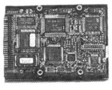
or

③●MF-2HD(5枚)
●シリコンキーボードカバー
もれなく!!サービス!!

※どちらかお選び下さい!!
(どっちが得かヨーク考えてネ!)

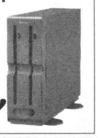
今月の推奨品(送料¥1,000)

■内蔵用ハードディスク
〈Compact XVI(CZ-674C)用〉
[KGU-HD80K]
Compact HD-80キット



定価¥168,000
限定特別価格¥TEL下さい!

■5インチフロッピーディスクユニット
〈X68000 Compact(CZ-674C-H)用〉
[CZ-6FD5]



定価¥99,800
限定特別価格¥TEL下さい!

周辺機器コーナー (送料¥500)

品名	定価	特価
●CZ-6BE2A 2MB RAM(CZ-634C/644C用)	(¥59,800)	▶特価¥42,500
●CZ-6BE2B 2MB RAM(CZ-634C/644C/674C用)	(¥54,800)	▶特価¥39,200
●CZ-6BE2D 2MB RAM(CZ-674C用)	(¥54,800)	▶特価¥39,000
●CZ-6BE2 2MB RAM	(¥79,800)	▶特価¥59,000
●CZ-6BE4C 4MB RAM	(¥98,000)	▶特価¥73,000
●CZ-6BFI 増設用RS-232ボード	(¥49,800)	▶特価¥35,800
●CZ-6BGI GP-IBボード	(¥59,800)	▶特価¥42,800
●CZ-6BMI MIDIボード	(¥26,800)	▶特価¥19,200
●CZ-6BNI スキャナ用パラレルボード	(¥29,800)	▶特価¥21,500
●CZ-6BPI 数値演算プロセッサボード	(¥79,800)	▶特価¥57,000
●CZ-6BOI ユニバーサルI/Oボード	(¥39,800)	▶特価¥29,800
●CZ-6EBI/BK 拡張I/Oボックス	(¥88,000)	▶特価¥66,000
●CZ-6VTI/BK カラーイメージ・ユニット	(¥69,800)	▶特価¥52,000
●CZ-8NM2A マウス	(¥6,800)	▶特価¥5,100
●CZ-8NTI マウストラックボール	(¥9,800)	▶特価¥7,300
●CZ-8NSI カラーイメージスキャナ	(¥188,000)	▶特価¥132,000
●CZ-6BCI FAXボード	(¥79,800)	▶特価¥57,000
●CZ-8TM2 モデムユニット	(¥49,800)	▶特価¥37,000
●LC-10CI-H カラー液晶ディスプレイ	(¥59,800)	▶特価¥45,800
●CZ-6TU GY/BK RGBシステムチューナー	(¥33,100)	▶特価¥23,800
●BF-68PRO 高性能CRTフィルター	(¥19,800)	▶特価¥14,500
●CZ-6MOI 光磁気ディスクユニット	(¥450,000)	▶特価¥330,000
●CZ-6BSI SCSIインターフェースボード	(¥29,800)	▶特価¥22,000
●CZ-6BL2 LANボード	(¥298,800)	▶特価¥219,000
●CZ-6BVI (ビデオボード)	(¥21,000)	▶特価¥15,400
●CZ-6BP2 数値演算プロセッサ	(¥45,800)	▶特価¥34,300
●AN-S100 スピーカーシステム(2本1組)	(¥36,600)	▶特価¥26,300
●JX-220X カラーイメージスキャナー	(¥168,000)	▶特価¥120,000

※クレジットの回数は1回〜60回、ボーナス併用などありますのでお電話でお問合せ下さい。

■本体セット:送料無料 (注)本体セット以外の周辺機器(プリンター、モデム、HDD等)及びソフトの送料は、北海道・九州地区=1ケロ¥1500、■その他離島地区は、1ケロ¥2000となります。

※上記料金には、消費税は含まれておりません。消費税が付加されますので、詳しくは、電話でお問合せ下さい。

■特に人気のある商品によっては、しばらくお待ち願うことがありますのでご了承下さい!!

■ビッグバーゲンセール実施中!!ゲームソフトへ(ビジネス)新製品続々入荷中!

■ボーナス〈2回・4回・6回〉払いOK!!手数料無料!!ご利用下さい。■店頭にて、新作ゲームソフト25〜30%OFF!!

X68000 ——堂々のラインアップ!!

■PROII/XVI■

ボーナス〈2回・4回・6回〉払いOK!!手数料無料!!

X68000 XVI エクシヴィ

(送料・消費税込)

超特価¥残念!表示不能!!

X68000XVI ドッカーン!プレゼント!!

—あなたのオクトから素敵な贈物—

今、XVIをお買い上げいただいた方は、プレゼントの①番か②番のどちらかお選び下さい。プラス③番はもれなくプレゼント!!

① F15ストライクイーグルII (定価¥10,800)

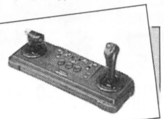
新作!大人気ゲームソフト!!

銀河英雄伝説II デラックスセット (定価¥12,600)

or

※どちらかお選び下さい!!

② インテリジェントコントローラ ■CZ-8NJ2 (CYBER STICK) シューティングゲーマーの必須アイテム!! (定価¥23,800)



\+

③ MD-2HD(10枚) シリコンキーボードカバー もれなく!!サービス!!

■**CZ-634C-TN**(定価¥368,000)

Ⓐ●CZ-634C-TN+CZ-606D-TN
定価合計¥447,800▶**超特価¥表示不能!**

12回	¥27,600	24回	¥14,600	36回	¥10,100	48回	¥ 8,000

Ⓑ●CZ-634C-TN+CZ-614D-TN
定価合計¥503,000▶**超特価¥表示不能!**

12回	¥31,200	24回	¥16,600	36回	¥11,500	48回	¥ 9,000

■**CZ-644C-TN**(定価¥518,000)

Ⓒ●CZ-644C-TN+CZ-606D-TN
定価合計¥597,800▶**超特価¥表示不能!**

12回	¥37,400	24回	¥19,800	36回	¥13,700	48回	¥10,800

Ⓓ●CZ-644C-TN+CZ-614D-TN
定価合計¥653,000▶**超特価¥表示不能!**

12回	¥40,900	24回	¥21,700	36回	¥15,000	48回	¥11,800

※クレジット表は、送料・消費税込!

X68000PROII ラストチャンス!! 〈BIGプレゼント付〉

(送料無料・税別)

X68000PROII (CZ-653C)

定価¥285,000
超特価¥138,000

新作!大人気ゲームソフト!!

銀河英雄伝説II デラックスセット (定価¥10,800)

プレゼント

さらに! ★JOY CARD(連射式)×2個
さらにさらに!! ★MD-2HD 10枚

■**CZ-653C**(定価¥285,000)

Ⓐ●CZ-653C+CU-21HD
定価合計¥433,000▶**超特価¥243,000**

12回	¥21,100	24回	¥10,500	36回	¥ 7,000	48回	¥ 5,200

Ⓑ●CZ-653C+CZ-606D
定価合計¥364,800▶**超特価¥198,000**

12回	¥17,100	24回	¥ 8,500	36回	¥ 5,700	48回	¥ 4,200

Ⓒ●CZ-653C+CZ-607D
定価合計¥384,800▶**超特価¥213,000**

12回	¥18,400	24回	¥ 9,200	36回	¥ 6,100	48回	¥ 4,600

Ⓓ●CZ-653C+CZ-614D
定価合計¥420,000▶**超特価¥236,000**

12回	¥20,400	24回	¥10,200	36回	¥ 6,800	48回	¥ 5,100

X68000ソフト大セール実施中!!(ゲームソフト25〜30%OFF) (送料¥500)

〈グラフィック〉●Z's STAFF PRO68K Ver.2.0 (シャフト)定価¥58,000 ……特価**¥36,500**

〈開発ツール〉●Cコンパイラ PRO-68K Ver.2.1 定価¥44,800 CZ-285LSD ……特価**¥32,500**

〈統合表計算ソフト〉BUSINESS PRO-68K Popular 定価¥28,000 CZ-286BSD ……特価**¥21,000**

〈レイアウト〉●Pressconductor PRO-68K 定価¥28,000 CZ-268BSD ……特価**¥21,000**

〈C言語〉●C & Professional Pack 定価¥58,000 ……特価**¥39,600**

〈音楽〉●Music studio PRO-68K Ver. 2.0 定価¥28,800 CZ-261MS ……特価**¥21,200**

〈CGシール〉●CANVAS PRO-68K 定価¥29,800 CZ-249GS ……特価**¥22,200**

〈ワープロ〉●Multiword Ver.1.1 定価¥32,000 CZ-225BSD ……特価**¥23,000**

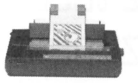
〈OS〉●OS-9 X68000 Ver.2.4 定価¥35,800 CZ-284SSD ……特価**¥26,900**

型名	商品	定価	特価
CZ-212BS	〈BUSINESS PRO-68K〉	(¥ 68,000)	¥ 48,000
CZ-213MS	〈MUSIC PRO68K〉	(¥ 18,800)	¥ 13,200
CZ-275MWD	〈SOUND SX-68K〉	(¥)	¥TEL下さい
CZ-215MS	〈Sampling PRO-68K〉	(¥ 17,800)	¥ 12,500
CZ-287SS	〈SX-WINDOW Ver.2.0〉	(¥ 12,800)	¥ 9,600
CZ-220BS	〈DATA PRO-68K〉	(¥ 58,000)	¥ 40,000
CZ-272CWD	〈Communication SX-68K〉	(¥ 19,800)	¥ 15,300
CZ-224LS	〈THE 福袋 V2.0〉	(¥ 9,900)	¥ 7,400
CZ-253BS	〈CARD PRO-68K Ver.2.0〉	(¥ 29,800)	¥ 20,800
CZ-258BS	〈Tlepotion PRO-68K〉	(¥ 22,800)	¥ 16,800
CZ-244SS	〈Homan 68K Ver.2.0〉	(¥ 9,800)	¥ 7,500
CZ-247MS	〈MUSIC PRO-68K〔MIDI〕〉	(¥ 28,800)	¥ 20,800
CZ-240BS	〈Stationery PRO-68K〉	(¥ 14,800)	¥ 11,500
CZ-243BS	〈CYBER NOTE PRO-68K〉	(¥ 19,800)	¥ 15,200

型名	商品	定価	特価
	〈Z's TRIPHY(デジタルクラフト)〉	(¥ 39,800)	¥ 27,000
	〈テラッツオ(ハミングバード)〉	(¥ 19,400)	¥ 13,600
	〈KAMIKAZE(サムシンググッド)〉	(¥ 68,000)	¥ 43,800
	〈Final Ver.3.2(エーエスピー)〉	(¥ 38,000)	¥ 29,000
	〈サイクロンEXPRESSα68〉	(¥ 98,000)	¥ 69,000
	〈G'ツール(ザインソフト)〉	(¥ 28,000)	¥ 18,600
	〈たーみのる2(SPS)〉	(¥ 17,800)	¥ 13,000
	〈G68K Ver.2 PRO〉	(¥ 22,000)	¥ 17,300
	〈C-TRACE68 Ver.3.0〉	(¥ 98,000)	¥ 68,500
CZ-251BS	〈ハイパーワード〉	(¥ 39,800)	¥ 29,400
CZ-260LS	〈XBAS to CHECKER PRO-68K〉	(¥ 9.800)	¥ 7,500
CZ-234LS	〈AI-68K〉	(¥188,000)	¥139,000
CZ-255GS	〈CANVASドローグラフィックLIB〉	(¥ 8,800)	¥ 6,600
CZ-256GS	〈CANVASドローグラフィックVol.2〉	(¥ 8,800)	¥ 6,600

プリンタ (送料¥1,000)

■CZ-8PC5-BK
熱転写カラー漢字
定価¥96,800
大特価**¥68,800**



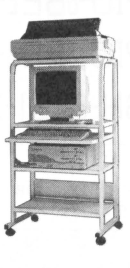
■IO-735X-B
カラーイメージジェット
定価¥248,000
大特価**¥154,000**

ハードディスク (送料¥1,000)

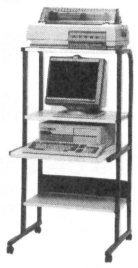
■システムサコム SCSI
- ●HD-J040 ¥ 89,000 42M/25ms‥大特価**¥ 61,000**
- ●HD-J100 ¥128,000 100M/20ms・大特価**¥ 87,000**
- ●HD-J130 ¥148,000 130M/20ms・大特価**¥101,000**
- ●HD-J170 ¥189,800 173M/20ms・大特価**¥123,000**

■ロジテック SCSI
- ●LHD-FM100E ¥ 99,800 100M………大特価**¥ 69,000**
- ●LHD-FM200E ¥138,000 200M/17ms・大特価**¥ 95,000**

パソコンラック〈送料無料〉



Ⓐ5段キャスター付
スライド式キーボード台
●1150(H)×640(W)×600(D)
定価¥38,000
特価**¥12,500**



Ⓑ4段キャスター付
●1250(H)×640(W)×700(D)
定価¥29,800
特価**¥8,800**

店頭新作ゲームソフト25〜30%OFF!! ビジネスソフト25%より特価中

★通信販売お申込みのご案内★ 〒144 東京都大田区蒲田4-6-7 TEL:03-3730-6271

お申込みはお電話でお願いします。お客様の〈住所〉〈氏名〉〈電話番号〉及び〈商品名〉をお知らせ下さい。●入金確認後ただちに商品をご送付いたします。

現金一括払い
銀行振込:お近くの銀行より(電信扱い)にてお振込み下さい。
現金書留:封筒の中に住所・氏名・商品名をご記入の上当社までお送り下さい。

クレジット
専用お申込用紙をお送り致します。ので、必要事項をご記入、ご捺印の上ご返送下さい。手続きは簡単です。

オクト ラクラク クレジット表

3回	3.5	6回	4.5	10回	6.0	12回	6.0
15回	9.0	18回	11.0	20回	12.0	24回	12.5
30回	17.0	36回	17.5	48回	23.0	60回	33.0

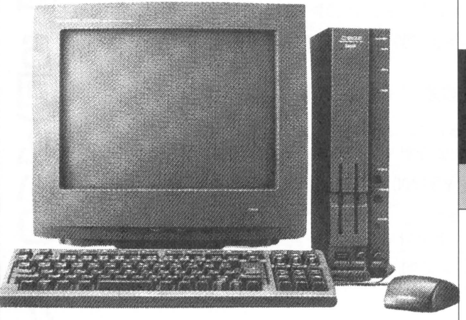
振込先
富士銀行 久ヶ原(クガハラ)支店 ㊜No.1824
三菱銀行 蒲田支店 ㊜No.0278691
株式会社 億人(オクト)

※掲載の価格は変動しますので、まずは、お電話にてご確認ください。
※上記料金には、消費税は含まれておりません。消費税が付加されますので、詳しくは電話でお問合せ下さい。
※銀行振込、または、現金書留でご注文の際には、あらかじめ電話でご確認の上、お申し込み下さい。

マイコンショップ川口

☎0482-25-1718

（消費税別）



New X68000
COMPACT XVI
~~¥298,000~~

CZ-674C-H…………¥298,000
CZ-608D-H…………¥ 94,800
AV-090-SC…………¥168,000

定価 ¥560,000
超特価

ソフト各種超特価ご奉仕中

CZ-219SS OS-9/X68000……定価¥29,800
CZ-213MS MUSIC PRO68K 定価¥18,800
CZ-214MS SOUND PRO68K 定価¥15,800
CZ-215MS Sampling PRO68K……定価¥17,800
CZ-220BS DATA PRO68K‥定価¥58,000
CZ-224LS The 福袋 Ver2.0‥‥定価¥ 9,980
CZ-225BS Multiword……定価¥32,000
CZ-251BS Hyper word……定価¥39,800

中古売買価格表

品名	買取り価格	売価
CZ-633C	160,000より	180,000より
CZ-644C	210,000より	230,000より
CZ-613C	105,000より	125,000より
CZ-603C	75,000より	95,000より
CZ-612C	85,000より	98,000より
CZ-602C	65,000より	85,000より
CZ-653C	75,000より	95,000より
CZ-663C	95,000より	115,000より
CZ-662C	75,000より	98,000より
CZ-652C	55,000より	75,000より
CZ-611C	70,000より	89,000より
CZ-601C	45,000より	65,000より
CZ-612D	35,000より	45,000より
CZ-602D	30,000より	39,800より
CZ-603D	20,000より	29,800より
CZ-604D	25,000より	34,800より
CZ-605D	45,000より	55,000より

プリンター

CZ-6VT1…………特価¥
CZ-8PG1…………特価¥
CZ-8PG2…………特価¥
CZ-8PK10…………特価¥
CZ-8NS1…………特価¥
CZ-6BC1…………特価¥
CZ-6BG1…………特価¥
CZ-6BP1…………特価¥
CZ-6BP2…………特価¥

ラムボード

CZ-6BE2A…定価¥59,800…特価¥
CZ-6BE2B…定価¥54,800…特価¥
CZ-6BE2D…定価¥　…特価¥
CZ-6BE1B…定価¥28,000…特価¥
CZ-6BE2……定価¥79,800…特価¥
CZ-6BE4C…定価¥98,000…特価¥
PIO-6BE1-A‥定価¥25,000…特価¥
PIO-6BE2-2M 定価¥50,000…特価¥
PIO-6BE4-4M 定価¥88,000…特価¥
SH-6BE1-1M‥定価¥25,000…特価¥

ファイル

CZ-6MO1…………定価¥450,000 特価¥
CZ-64H…………定価¥120,000 特価¥
CZ-68H…………定価¥160,000 特価¥

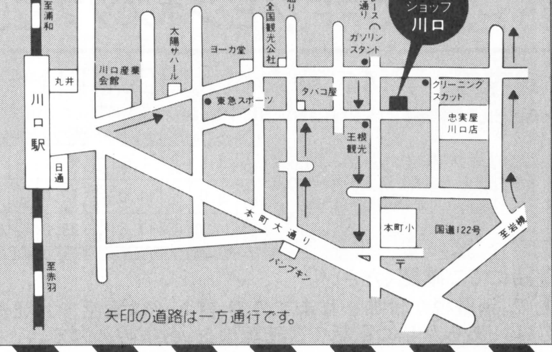
その他機種

CZ-8NS1 カラーイメージスキャナ‥定価¥188,000 特価¥
JX-220X カラーイメージスキャナ…定価¥168,000 特価¥
CZ-6BN1 スキャナ用パラレルボード‥定価¥ 29,800 特価¥
CZ-6VT1 カラーイメージユニット‥定価¥ 69,800 特価¥
CZ-6BV1 ビデオボード…………定価¥ 21,000 特価¥
CZ-8TM2 モデムユニット…………定価¥ 49,800 特価¥
CZ-8NJ2 インテリジェントコントローラ…………定価¥ 23,800 特価¥
CZ-8NM3 マウス・トラックボール‥定価¥ 9,800 特価¥
CZ-8NT1 トラックボール…………定価¥ 6,888 特価¥
CZ-8NJ1 ジョイカード…………定価¥ 1,700 特価¥
CZ-6BC1 FAXボード…………定価¥ 79,800 特価¥
CZ-6BM1A MIDIボード…………定価¥ 26,800 特価¥
CZ-6BP1 数値演算プロセッサ……定価¥ 79,800 特価¥
CZ-6BP2 数値演算プロセッサ……定価¥ 45,800 特価¥
CZ-6TU-BK-GY RGBシステムチューナー 定価¥ 33,100 特価¥

★クレジット回数1～60回まで設定自由

回数	1	3	6	12	15	20	24	36	42	48	54	60
金利(%)	2.5	3.5	4.5	6	9	12	12.5	17.5	22	23	28.5	29.5

中古品も取扱っております。



矢印の道路は一方通行です。

通信販売をご利用の方

―全国通販―

通信販売をご利用の方は、売値の変動がありますので在庫、値段をあらかじめ確認のうえ電話で、商品名及びお客様の住所・氏名・電話番号をお知らせ下さい。

X68000 XVIシリーズ大特価セール！

AIBIT アイビット電子株式会社

8月末迄

X68000専用ハードディスク

HXD-040 特価¥59,000

HXD-042 特価¥64,000

HXD-140（内蔵用） 特価¥65,000

対応機種　CZ-600C　CZ-602C　CZ-652C　CZ-601C　CZ-603C　CZ-653C

★新　発　売★

●書院パソコン

PC-WD1A	¥330,000
PC-WD1AD	¥450,000
PC-WD1B	¥430,000
PC-WD1BD	¥590,000

●ハイパー電子マネージメント手帳

PV-F1	¥128,000

I

CZ-674CH	(本体)	¥298,000
CZ-608DH	(0.28㎜ディスプレイ)	¥ 94,800
CZ-6FD5	(XVI用外付け 5インチ2ドライブ)	¥ 99,800
標準価格		¥492,600
35%off	特価	**¥320,000**

II

CZ-674CH	(本体)	¥298,000
CZ-606D	(0.31㎜ディスプレイ)	¥ 79,800
CZ-6FD5	(XVI用外付け 5インチ2ドライブ)	¥ 99,800
標準価格		¥477,600
35%off	特価	**¥310,000**

III

CZ-674CH	(本体)	¥298,000
CZ-608DH	(0.28㎜ディスプレイ)	¥ 94,800
標準価格		¥392,800
30%off	特価	**¥278,000**

IV

CZ-674CH	(本体)	¥298,000
CZ-606D	(0.31㎜ディスプレイ)	¥ 79,800
標準価格		¥377,800
30%off	特価	**¥268,000**

他周辺機器及びポケコン全機種取り扱い。

全商品新品完全保証付〉

シャープ・シャープ周辺機器（拡張機器全機種、プリンター他）・富士通・NEC常時取り扱い。
シャープ・カシオポケコン全機種取り扱い。PACIFIC・YHP・キヤノンも取り扱い。
学校、企業納入受け賜ります。送料別料金。★上記商品価格には、消費税は含まれておりません。
特価表及び資料をご希望の方は、72円切手を同封の上お送りください。

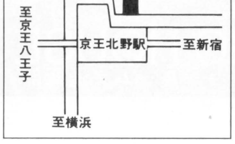
通信販売のお問い合せ、御注文は

TEL.0426-45-3001(本店)　FAX.0426-44-6002

営業時間/10:00～19:00●電話受付/9:00～22:00迄可●定休日/水曜日

SHARP SUPER EXE SHOP

アイビット電子株式会社　〒192　東京都八王子市北野町560-5



上記の広告商品はすべて店頭販売もしております。

全国通信販売
北海道から沖縄まで

★送料はご注文の際にお問い合わせ下さい。
★掲載の商品は、すべて新品、保証書付きです。
★掲載の商品は充分用意してありますが、ご注文の際は、在庫の確認の上、現金書留または、銀行振込でお申し込み下さい。全商品クレジットでも扱っております。
★お申し込みの際は必ず電話番号を明記して下さい。
★商品、品切れの節はご容赦下さい。

富士銀行八王子支店　（普）1752505

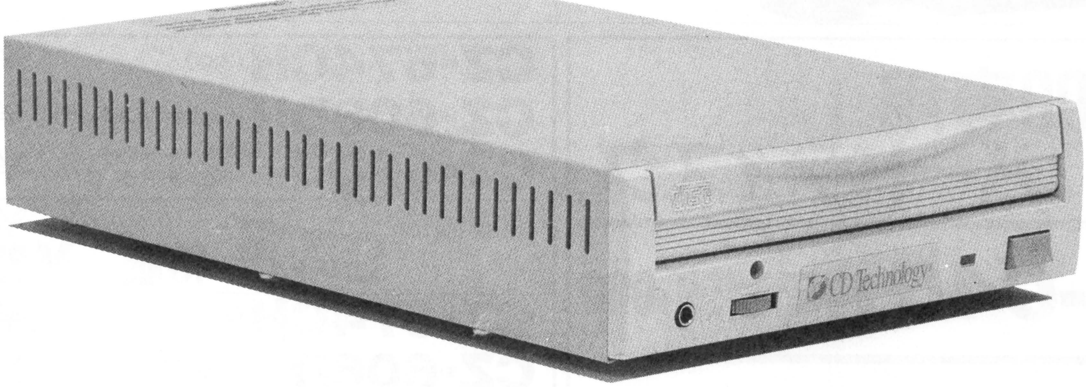
CD-ROM Drive
for
X68000
マルチメディアへの誘い

X68000 Pro SHOP
BASIC HOUSE
KEISOKUGIKEN Corp.
TEL 0286-22-9811　FAX 0286-25-3970

FirstClassTechnology制作のCD-ROM Device Driverを付属させ、ついにX68000用CD-ROM Driveの登場です。本製品を使用することにより、MS-DOSやPC-9801シリーズ、FM-TOWNSなどで採用されている、ISO9660規格のCDをHuman68K/SX-WINDOWで直接扱えるようになります。

また、将来の拡張にも柔軟に対応できるSCSIインターフェースによる接続を採用。ディジーチェーンによって既存のSCSIハードディスクとの同時使用も可能です。

担当　登坂高明



ドライブ仕様

型番	KGU-XCD
使用ドライブ	東芝　XM-3301
平均アクセスタイム	325mSEC
インターフェース	SCSI
キャッシュメモリー	64KB
オーデオ出力	RCA-Phono端子×2 ステレオヘッドホン端子
電源	専用ACアダプター
外形寸法	150×228×50（電源部含まず）

付属サポートソフト

ISO9660準拠デバイスドライバ
MusicPlayer for SX-Window
Macintosh™用ファイルビューア　for SX-Window

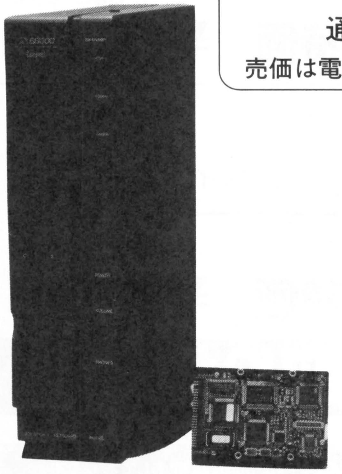
KGU-XCD対応
X68000 CD-ROM第一弾！「フリーウェア集」
Free Soft Ware Selection - CD68K
近日発売
CD-ROM広辞苑検索ユーティリティ for SX-WINDOW
近日発売

標準価格¥118,000-

※Macintosh™はAppleComputerの登録商標です

大好評発売中！
2.5inch 80MB HDを内蔵
X68000CompactHD

標準価格　¥466,000
通販特価販売中
売価は電話にてお問い合わせください！



ドライブ
Quantum Go-80s™
容量
62MByte
アクセスタイム
16mSEC
インターフェース
SCSI
外部SCSI機器との同時使用可

低金利クレジット　通信販売送料 全国一律￥1,000 長期クレジット可能
※表示価格に消費税は含まれておりません

株式会社 計測技研 マイコンショップ BASIC HOUSE
本社／ショールーム／通販部
〒321 栃木県宇都宮市竹林町503-1
TEL 0286-22-9811　FAX 0286-25-3970

OS-9 Ver.2.4 対応パッケージリリース

microware

OS-9/X68000 C&Professional Pack. V3.2

C&プロフェッショナル・パッケージは、OS-9/X68000上で動作するマイクロウェア・Cコンパイラとユーティリティ・ソフトのパッケージです。

◆マイクロウェア・Cコンパイラの特徴

他OSの、K&R準拠、ANSI準拠のアプリケーション、あるいはUNIX上のアプリケーションは、特に修正することなく容易に移植できます。

このコンパイラはCPUのインストラクションに最適化されており、生成されるオブジェクトが、最小、最速になるように複数レベルに渡るオプティマイズを実行しています。

◆拡張機能

1. シンボリック・デバッグ
2. 強力なエラー診断機能
3. 高速なコンパイル・スピード
4. 豊富なオプション
5. アセンブリ言語とのインタフェース
6. OS-9/X68000用拡張ライブラリ

※バージョンアップサービスを予定しておりますので、お早めにユーザ登録をお済ませ下さい。

◆付属ユーティリティ・ソフト

●SrcDbg(ソース・レベル・デバッガ)

SrcDbgは、C言語で書かれたプログラムのテストやデバッグをソース・レベルで行うユーティリティです。

●μMACS(マイクロマックス)

μMACSは、UNIX上で広く利用されているスクリーンエディタ"EMACS"のOS-9版サブセットです。

◆パッケージ内容

マイクロウェア・Cコンパイラ
標準ライブラリ
OS-9/X68000専用ライブラリ
ヘッダ・ファイル
OS-9/X68000専用ヘッダ・ファイル
アセンブラ
リンカ
ユーザステート・シンボリック・デバッガ
ソース・レベル・デバッガ
漢字フル・スクリーン・エディタ

◆付属マニュアル

Cコンパイラ・ユーザーズ・マニュアル
アセンブラ・リンカ・デバッガ・ユーザーズ・マニュアル
ソース・レベル・デバッガ・ユーザーズ・マニュアル
μMACSユーザーズ・マニュアル
OS-9/X68000専用ライブラリ・マニュアル1,2

3.5"2HD 5"2HD 2枚組
定価¥80,000

OS-9/X68000 テクニカル・デベロップメント・キット Technical Development Kit V2.4

OS-9/X68000テクニカル・デベロップメント・キットには、OS-9上でのプログラミングのためのマニュアルとシステム・ステートでのデバッグを可能とするデバッガが含まれています。

また、デバイス・ドライバ作成のために、※各種サンプルソースコードが付属しています。

※サンプルソースコードに関してのお問い合わせはご遠慮願います。

◆パッケージ内容

マニュアル
　システムコール
　テクニカル
　I/Oテクニカル
　システムステート・デバッガ・ユーザーズ
　ROMデバッガ・ユーザーズ
ソフトウェア
　SysDbg
　RomBUG
※各種サンプル・ソースコード

◆システムステート・デバッガ(SysDbg)

SysDbgは、OS-9システムの拡張など、I/Oドライバの開発を強力に支援するシンボリック・ステート・デバッガです。

◆ROMデバッガ(RomBUG)

RomBUGは、OS-9とは独立したデバッガです。起動時に必要なコンソールやディスクなどのデバイス・ドライバをデバッグすることができます。

供給メディア
3.5"2HD 5"2HD
定価 ¥38,000円

＊会社名・製品名は、各社の商標または登録商標です。
※製品の内容等は予告なく変更されることがあります。

OS-9/X68000はシャープ㈱から販売しています。

マイクロウェア・システムズ株式会社
〒101 東京都千代田区外神田2-17-3 代表(03)3257-9000 Fax(03)3257-9200

SHARP

コンピューター事業拡張につき プログラマー募集！

提供するのは、X68000の才能をひき出す仕事です。

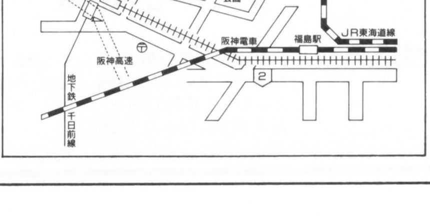
勤務地 大阪・東京・岡山
(男女不問・現地面接可)

■会社概要
設　　立■昭和44年
資 本 金■1,500万円
従業員数■17名
平均年齢■26歳
■事業内容
パーソナルコンピュータ・AXによる自社ソフトパッケージの開発及びオーダーメイド販売サポート
X68000による画像作成業務
資　　格■高卒以上30歳位迄の方
※未経験者歓迎
給　　与■経験・能力等与慮の上、当社規定により優遇いたします。例 25歳 ㊊ 176,000円
※別途報奨金制度あり
待　　遇■昇給年1回・賞与年2回 手当/業務・営業・皆勤 交通費全額支給
勤務時間■9:00～18:00
福利厚生■各種社会保険完備 退職金制度 財形貯蓄制度 社内旅行有

経験の有無を問わず、X68000大好き人間 歓迎。経験者には、実力を発揮する場を、未経験者には丁寧な指導をお約束します。
シャープ、XEROX等のシステム機器販売から、シャープ・コンピューターのシステムプレゼンテーターとしてメーカーの期待を担う当社で活躍して下さい。

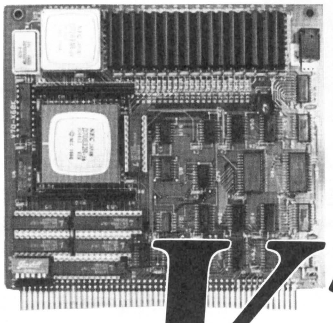
株式会社ラインシステム
本社 〒553 大阪市福島区鷺洲3丁目1 TEL06-458-7313 担当 菊田
〒115 東京都北区浮間3-2-16 エスポワール403 TEL03-5994-2087 担当 鈴木

休日休暇■隔週休2日制(完全週休2日制も検討中)
祝日
有給・特別・夏期・年末年始休暇等
応　　募■電話連絡の上、履歴書(写真貼付)を持参又は郵送して下さい。追って詳細を連絡いたします。
※入社日相談に応じます。
※応募の秘密厳守いたします。
交　　通■阪神、地下鉄野田駅下車 徒歩7分



ACCESS

for X68000



V70アクセラレータの魅力を探る

V70 アクセラレータ

高速処理を実現

V70(μPD70632)は、日本電気(株)が開発した32ビットCMOSマイクロプロセッサである。このマイクロプロセッサは、数々の高度な特徴を備えており、いわゆるマイクロプロセッサのカテゴリーとしては、CISCに属する。V70アクセラレータは、このCISCチップを利用したハードウェアとしては最高峰に位置するものである。

また、V70は、それ自身浮動小点演算機構を内蔵しており、高速演算が可能であるが、更に高速、高精度な演算を行う目的で、アドバンスト・フローティング・ポイント・プロセッサ(AFPP)が標準で搭載されている。このAFPPには、右表に挙げるような特徴があり、非常に魅力的なチップなのである。

たとえばコンピュータグラフィックス等、高度な処理を要求されるシーンで、その威力を充分に発揮する。V70アクセラレータで、きみのX68000がスーパーワークステーションへと生まれ変わるのだ。

簡単に利用できる

通常アドオンCPUボードを利用する場合、そのCPUにプログラムを実行させるのもなかなかたいへんである。たとえばV70CPUにプログラムを実行させるには、まず、V70側にリセットをかけ、X68000より共有RAMの最上位アドレス部にV70側のスタートアッププログラムをロードし、リセットを解除する。V70CPUは、0FFFFFFF0Hより実行を開始する。もちろん、この後V70アクセラレータとX68000の間で適切なやりとりをして、目的とするプログラムをV70アクセラレータのローカルRAMエリアにロードし、実行して行かなければならない。

本来ならば以上のような手順をとらなければならないが、通常、ユーザはここで説明したような操作を行う必要はない。なぜならば、付属のシステムモニタ、コマンドシェルが、そのようなやりとりをすべて行ってくれるからである。

開発環境の充実

アセンブラ・リンカはもちろん、開発の強力な味方であるソースコードデバッガやシステムモニタ、さらにはフロートエミュレータ・コマンドシェルまでついている。32ビットマイクロプロセッサV70の特徴である仮想記憶、メモリプロテクション、CPUレベルでのデバッグ機能などをサポートしている。おまけにCコンパイラはというと、Human68k上のCコンパイラと互換性が高く、プログラムをほとんど修正なしで実行できてしまうのである。

AFPPの特徴

○コプロセッサ・プロトコル内蔵
- ●V70対応

○高速数値演算
- ●システム・クロック:16MHz(最大 20MHz)
- ●四則演算(倍精度):0.9MFLOPS*(CPU:V70、20MHz動作時)
- ●三角関数演算(倍精度):6.0μsec(CPU:V70、20MHz動作時)
- ●ベクトル/行列演算(倍精度):5.8MFLOS*(CPU:V70、20MHz動作時)

○IEEE754標準に準処

○豊富な命令セット
- ●転送/変換命令
- ●浮動小数点演算命令(四則、比較、その他)
- ●制御命令
- ●初等関数演算命令(三角、逆三角、指数、対数、双曲線)
- ●ベクトル/行列演算命令

○5種類のデータ・フォーマット
- ●2進実数:32ビット、64ビット、80ビット
- ●2進正数:32ビット、64ビット

○32本のデータ・レジスタ内蔵(80ビット幅)

○CPUと並行に動作可能

○68ピンPGAパッケージ

アセンブラ

●数百におよぶ命令セット、20種類におよぶアドレシングモードすべてサポート。

●コプロセッサ命令をフルサポート。
1命令で浮動小数点演算が可能。

システムモニタ

●仮想メモリモードを採用。
16MByteのメモリ空間をサポート。
大きなアプリケーションでも実行可能。
(同時使用可能メモリ2MByteまで)。

●X68000のIOCSやHuman68kとほぼ同時のシステムコールが利用可能。

ソースコードデバッガ

●コンソールモード、リモートモード、フルスクリーンモードの3つの画面モードを持つ。
状況に合わせたデバッグが可能。

●C言語のソースレベルでのデバッグをサポートし、C言語レベルでの式の評価、行単位、関数単位でのデバッグ可能。

フロートエミュレータ

●Human68k上の従来のアプリケーションを変更せずに、そのまま高速な浮動小数点演算が可能。

コマンドシェル

●V70用アセンブラ、コンパイラなどで記述されたV70の実行プログラムを、Human68kの実行形式プログラムを実行するのと同様の感覚で実行する環境を提供。

価格

●ボードパッケージ (XVI対応)
VDTK-X68K…………¥248,000

●オプションソフト(Cコンパイラ)
VDTK-C-X68K………¥68,000

購入方法 上記商品は当面の間、通信販売とさせて頂きます。
購入ご希望の方は、住所、(社名、所属)氏名、電話番号をお知らせ下さい。
注文書をお送りいたします。

《オプション》Cコンパイラ

V70アクセラレータ用のC言語で開発するためのCコンパイラ。
C標準ライブラリの他、X68000本体のシステムコールを利用するための、DOSコールライブラリやIOCSコールライブラリも用意。

※製作:ボード…………有限会社アクセス
ソフトウェア……株式会社ハドソン

有限会社アクセス 〒101 東京都千代田区神田神保町1-64
神保町協和ビル7F
☎03(3233)0200(代) FAX.03(3291)7019

資料請求券
oh!X
8月号

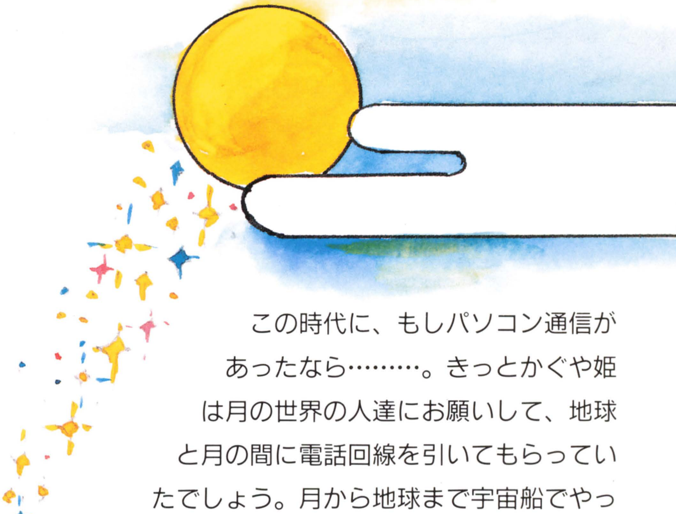
パソコン/ワープロ通信ネットワークサービス

J&P HOT LINE

タイムトラベルシリーズ

第5回

《もし、この時代にパソコン通信があったなら》



地球への思いは断ちがたく

かぐや姫

日本最古のＳＦと呼ばれている竹取物語。
輝く竹の中から生まれたかぐや姫は、成人して美しい女性となり、
５人の男性と帝からプロポーズを受けます。
やはり月からの異邦人。地球人にはない、
特別の美しさをもっていたのかもしれませんね。
でも、かぐや姫は５人の男性からの贈り物にも
帝からの手紙にも喜ばず、ただ悲しそうな顔をするだけ。
実は月に帰る日が近づいて、
自分を育ててくれたおじいさん、おばあさんと
お別れするのが辛かったんですね。

この時代に、もしパソコン通信があったなら………。きっとかぐや姫は月の世界の人達にお願いして、地球と月の間に電話回線を引いてもらっていたでしょう。月から地球まで宇宙船でやって来れた人達なら、その程度のことならきっとできたはず。月と地球はひとつの街として結ばれていたかもしれません。かりに、かぐや姫が月に帰っても、まるで隣街で暮らしているように毎日の出来事をおじいさん、おばあさんに話していたでしょうし、プロポーズした男性たちも、月でのかぐや姫の暮らしを知ることができたはずです。でも、そうなるとあの悲しいラストシーンは、「じゃ、いってきまーす！」というかぐや姫の明るい声でしめくくられ、後世に残る名シーンにはならなかったかもしれませんね。

パソコン通信なら、こんな楽しさ。

月の世界との交信とまではいきませんが、地球サイズでネットワークが広がるパソコン通信。あなたも、自分の世界が大きく広がるパソコン通信のネットワークの中に飛び込んでみませんか？

J&P HOT LINEへの
ご入会はスタータキットで。

買ったその日から
2週間無料で
アクセスできます。

お求めは、下記のお店へ。又は現金書留にて、¥3,000＋¥90(消費税3%)＝¥3,090を事務局までお送り下さい。
すぐにスタータキットをお送りします。

お問い合わせは
〒556 大阪市浪速区日本橋西1-6-5 上新電機株式会社
J&P HOTLINE事務局宛　TEL.(06)632-2521

スタータキットのお求めはJ&P各店でどうぞ。

渋谷店	東京都渋谷区道玄坂2丁目28番4号	☎(03)3496-4141
町田店	東京都町田市森野1丁目39番16号	☎(0427)23-1313
八王子店	東京都八王子市旭町1番1号八王子そごう7F	☎(0426)26-4141
立川店	東京都立川市幸町4－39－1	☎(0425)36-4141
三鷹店	三鷹市野崎1－20－17	☎(0422)31-6251
横浜店	横浜市西区北幸2-9-5横浜HSビル1F	☎(045)313-6711
焼津インター店	静岡県焼津市越後島385	☎(054)626-3311
富山店	富山市掛尾町300番地	☎(0764)22-5033
金沢店	金沢市入江2－63	☎(0762)91-1130
寺地店	金沢市寺地2－3	☎(0762)47-2524
大須店	名古屋市中区大須4丁目2-48	☎(052)262-1141
テクノランド	大阪市浪速区日本橋5丁目6番7号	☎(06) 634-1211

メディアランド	大阪市浪速区日本橋5丁目8番26号	☎(06) 634-1511
コスモランド	大阪市浪速区難波中2丁目1番17号	☎(06) 634-3111
U. S. LAND	大阪市浪速区日本橋4丁目9番15号	☎(06) 634-1411
ビジネスランド	大阪市北区梅田1-1-3大阪駅前第3ビルB2	☎(06) 348-1881
高槻店	高槻市高槻町11番16号	☎(0726)85-1212
くずは店	枚方市楠葉花園町15番2号	☎(0720)56-8181
千里中央店	豊中市新千里東町1-3 SENCHU PAL 2番街4F	☎(06) 834-4141
摂津富田店	高槻市大畑町24－10	☎(0726)93-7521
寝屋川店	寝屋川市緑町4－20	☎(0720)34-1166
枚方バイパス店	枚方市田口3－41－7	☎(0720)48-1211
藤井寺店	藤井寺市岡2丁目1番33号	☎(0729)38-2111
岸和田店	岸和田市土生町2451－3	☎(0724)37-1021

さんのみや1ばん館	神戸市中央区八幡通3-2-16	☎(078)231-2111
西宮店	兵庫県西宮市河原町5－11	☎(0798)71-1171
伊丹店	伊丹市昆陽池1－63	☎(0727)77-5101
姫路店	姫路市東延末1丁目1番住友生命姫路南ビル1F	☎(0792)22-1221
京都寺町店	京都市下京区寺町通仏光寺下ル恵比須之町549	☎(075)341-4411
京都近鉄店	京都市下京区烏丸通七条下ル東塩小路町702	☎(075)341-5769
和歌山店	和歌山市元寺町4丁目4番地	☎(0734)28-1441
和歌山南店	和歌山市中島368	☎(0734)25-1414
学園前店	奈良市学園北1－8－10	☎(0742)49-1411
奈良1ばん館	奈良市三条町478－1	☎(0742)27-1111
新大宮店	奈良市法華寺町83－5	☎(0742)35-2611
郡山インター店	大和郡山市横田693－1	☎(07435)9-2221
熊本店	熊本市手取本町4－12	☎(096)359-7800

SHARP

目の付けどころが、シャープでしょ。

このサイズは、まさにX68000の本来的にもつ創造力に、
無限大の可能性をひらくことになるだろう。

X68000
PERSONAL WORKSTATION・XVI
Compact

New

本体＋キーボード＋マウス
2HD3.5インチFDDタイプ CZ-674C-H(グレー) 標準価格298,000円(税別)
14型 カラーディスプレイ(ドットピッチ0.28mm)
CZ-608D-H(グレー) 標準価格94,800円(税別)

なにが生まれるか、夢を抱いて触れてください。体積比44%のコンパクトなボディに鋭さと優しさがギッシリ詰まっています。

■CZ-674C SPECIFICATION ●MPU：68000(16MHz/10MHz)●メインメモリ：2MB(本体内に8MB、最大12MBまで拡張可能)●表示エリアサイズ：テキスト/1024×1024ドット・4プレーン、グラフィック/1024×1024ドット・4プレーン(各512×512ドット・16プレーン)●表示画面モード：テキスト/実画面エリア1024×1024ドットのとき■高解像度モード＝768×512ドット・640×480ドット・512×512ドット・512×256ドット・256×256ドット■標準解像度モード＝512×256ドット・256×256ドット・(512×512ドットインターレース)、各モードともドットごとに65,536色中16色指定可能、グラフィック/実画面エリア1024×1024ドットのとき■高解像度モード＝768×512ドット・640×480ドット・512×512ドット・512×256ドット・256×256ドット■標準解像度モード＝512×256ドット・256×256ドット・(512×512ドットインターレース)、各モードともドットごとに65,536色中16色指定可能/実画面エリア512×512ドットのとき■高解像度モード＝512×512ドット・512×256ドット・256×256ドット■標準解像度モード＝512×256ドット・256×256ドット・(512×512ドットインターレース)、各モードとも①ドットごとに65,536色から任意の色指定可能(1面)②ドットごとに65,536色中256色指定可能(2面)③ドットごとに65,536色中16色指定可能(4面)●スプライト：パターン定義/サイズ＝16×16ドット/パターン、定義数＝128パターン(バックグラウンド2面未使用時最大256パターン)、色＝1パターンにつき65,536色中16色(ドット単位)、/座標系＝1024×1024ドット、表示画面＝水平512ドットor256ドット、垂直512ラインor256ライン、表示制限＝128スプライト/画面、32スプライト/ライン●特殊機能：スムーズスクロール・プライオリティ機能・パレット機能・半透明機能・実画面スクロール機能・スーパーインポーズ機能●サウンド機能：FM音源/2ch、8オクターブ、8重和音同時出力、音声合成/AD PCM(Adaptive Differential PCM)●フロッピーディスクドライブ：1.2MB・2HD・3.5インチフロッピーディスクドライブ(オートイジェクト機能)2基搭載●入力装置(同梱)：マウス、ASCII準拠キーボード●インターフェイス：プリンタ(セントロニクス社仕様に準拠)、ジョイスティック(2個)、アナログRGB出力、オーディオ入出力、RS-232C、外部フロッピーディスク、マウス、イメージ入力、SCSI、キーボード●専用ソケット：増設RAM用ソケット●拡張I/Oスロット：2スロット内蔵(10MHz駆動)●OS・言語：Human68k、X-BASIC、SX-WINDOW●消費電力：定格26W(最大56W・待機時4W以下)●動作温度・湿度範囲：10℃～35℃・35%～75%●外形寸法・重量：本体/幅78×高さ330×奥行き260mm・4.2kg、キーボード/幅380×高さ38×奥行き170mm・0.95kg、マウス/幅63×高さ37×奥行き97mm・0.11kg●付属ソフト：SX-WINDOW ver.2.0、Human68k ver.2.0、X-BASIC ver.2.0、辞書ver.2.0、日本語ワードプロセッサ ver.1.1ほか。

●5.25インチ増設用フロッピーディスクドライブ CZ-6FD5 標準価格99,800円・税別〔接続ケーブル同梱〕●ディスプレイテレビ/CZ-6TU用RGBケーブル CZ-6CR1 標準価格4,500円・税別
●ディスプレイテレビ/CZ-6TU用テレビコントロールケーブル CZ-6CT1 標準価格5,500円・税別●SCSI変換ケーブル CZ-6CS1 標準価格12,000円・税別

●お問い合わせは…電子機器事業本部システム機器営業部〒545大阪市阿倍野区長池町22番22号☎(06)621-1221(大代表) 電子機器事業本部AVCシステム事業推進室〒162東京都新宿区市谷八幡町8番地☎(03)3260-1161(大代表) シャープ株式会社

T1002179080608 雑誌 02179-8

Oh!X 8月号 1992年8月1日発行(毎月1回1日発行) 通巻124号 昭和58年11月2日第三種郵便物認可 ソフトバンク株式会社発行 Printed in Japan 定価800円(本体